An Argus Specialist Publication

October 1985

£1.00

Your COMMODORE

YOUR BEST INDEPENDENT COMMODORE MAGAZINE



Build a better

Going Forth – a round up of packages for your 64

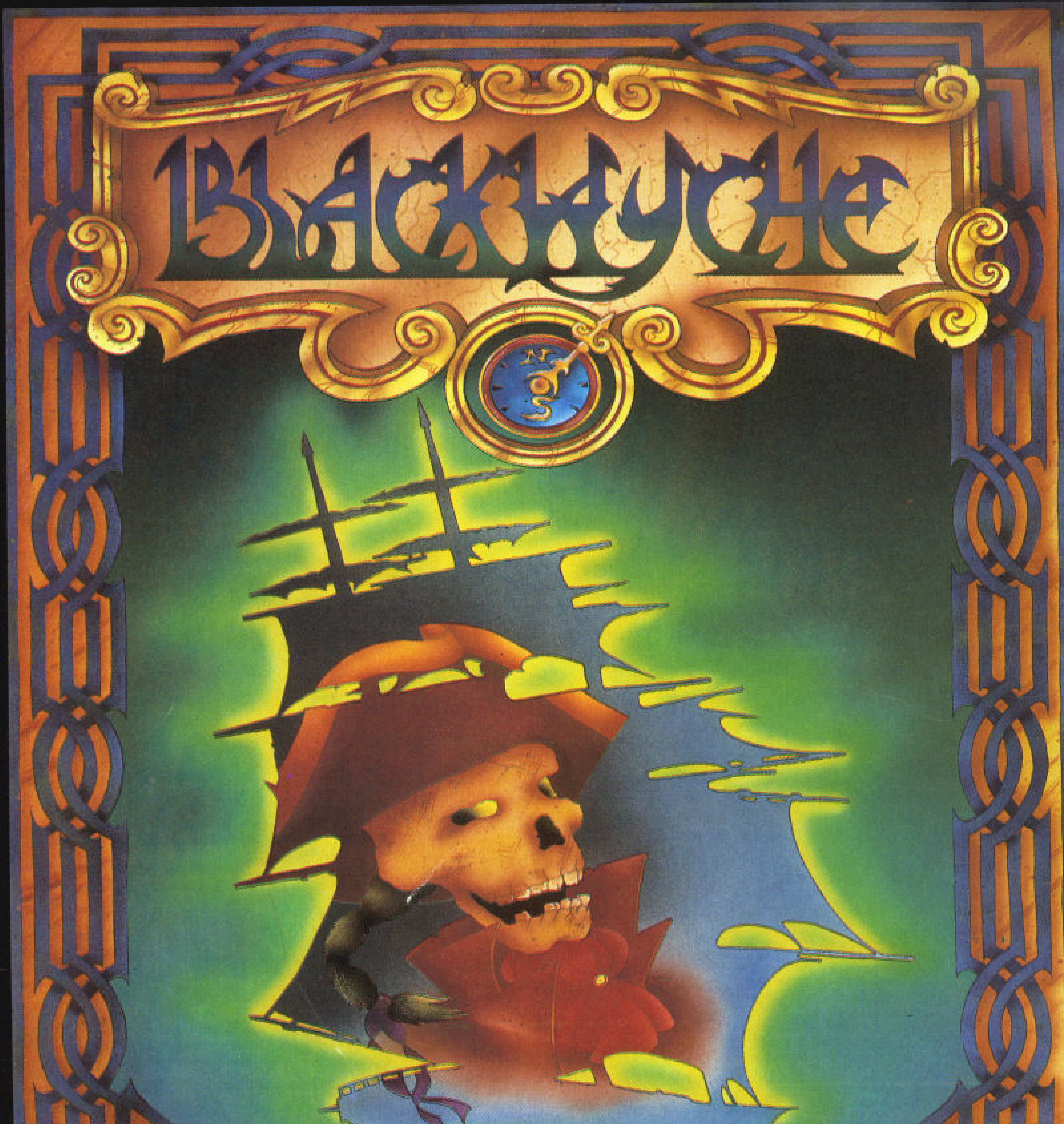
Operation C-16: Complete memory map

Crowther's latest blockbuster – reviewed

Competition – 30 copies of William Wobbler must be won

FIRST BIRTHDAY EDITION

COMMODORE 64





THE ACTION-PACKED COMBAT SIMULATION



◄ *OUT OF YOUR COCKPIT WINDOW AS YOU SWOOP DOWN TO TAKE OUT SOME TANKS*

'SKYFOX' is the most realistic awe-inspiring combat simulation you've ever seen on your computer. *'SKYFOX' special features include —*

- ► **AIR TO AIR & AIR TO GROUND COMBAT**
- ► **3D SCROLLING FLIGHT SIMULATION**
- ► **15 SCENARIOS** (from training mission to massive invasion)
- ► **5 SKILL LEVELS** (from cadet to ace of the base)
- ► **AUTO PILOT FEATURE FOR LAUNCH & ENEMY SEARCHES**
- ► **ON BOARD & BASE TACTICAL-COMBAT COMPUTERS**
- ► **CONTINUOUS LASER CANNONS**
- ► **HEAT-SEEKING & GUIDED MISSILES**
- ► **JOYSTICK & KEYBOARD CONTROLS**



ENEMY PLANES ATTACK SUDDENLY WITH DEADLY FORCE ►

"SKYFOX" NOW • GET "SKYFOX" NOW • GET "SKYFOX"

Available for the Apple II, II+, IIc, IIe, & Commodore 64/128 on Disk, and NOW CBM 64 Cassette.

Available from all good software retailers *— if its not there, please order it — or in case of difficulty send your crossed cheque/P.O. made out to* ***Ariolasoft U.K. Ltd.,*** *including your own name and address, to Ariolasoft U.K. Ltd., Suite 105/106, Asphalte House, Palace Street, London SW1E 5HS.*

► **SKYFOX** — *Commodore Cassette* **£9.95,** *Disk* **£12.95** ► **SKYFOX** — *Apple Disk* **£17.95**

ariolasoft
HIGH PERFORMANCE PROGRAMS



VOLUME 2 NUMBER 1

Editor: Stuart Cooke
Assistant Editor: Marie Curry
Software Assistant: John Donovan
Advertisement Manager: Mike Segrue
Advertisement Copy Control: Laura Champion
Group Editor: Wendy J. Palmer
Publishing Director: Peter Welham
Origination: Ebony Typesetting
Design: MM Design

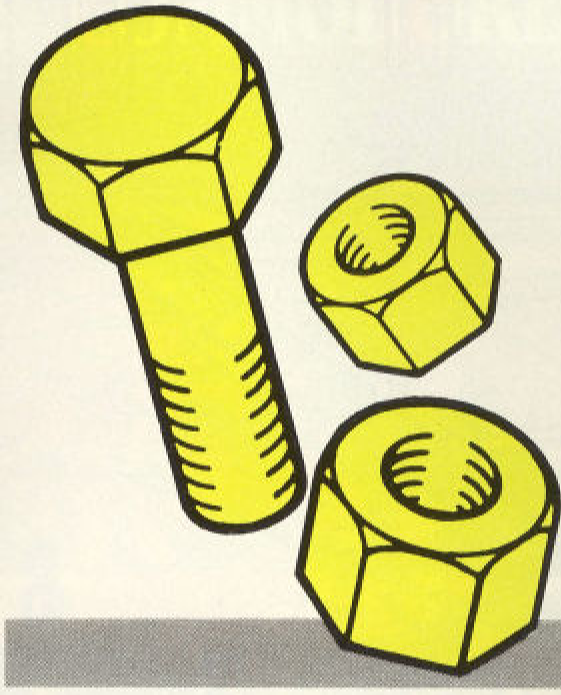
Editorial & Advertisement Office
No 1 Golden Square,
London W1R 3AB
Telephone: 01-437 0626
Telex: 8811896

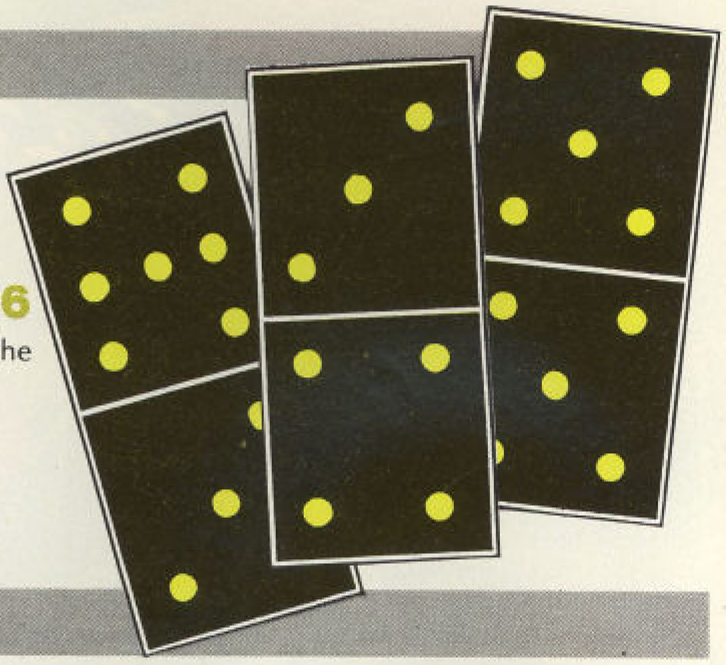
Your Commodore is a monthly magazine appearing on the first Friday of each month.

Distribution by: Argus Press Sales & Distribution Ltd, 12-18 Paul Street, London EC2A 4JS.
Printed by: Alabaster Passmore & Sons Ltd, Tovil, Maidstone, Kent.

Subscription rates upon application to Your Commodore Subscriptions Department, Infonet Ltd, Times House, 179 The Marlowes, Hemel Hempstead, Herts. HP1 1BB.

FEATURES





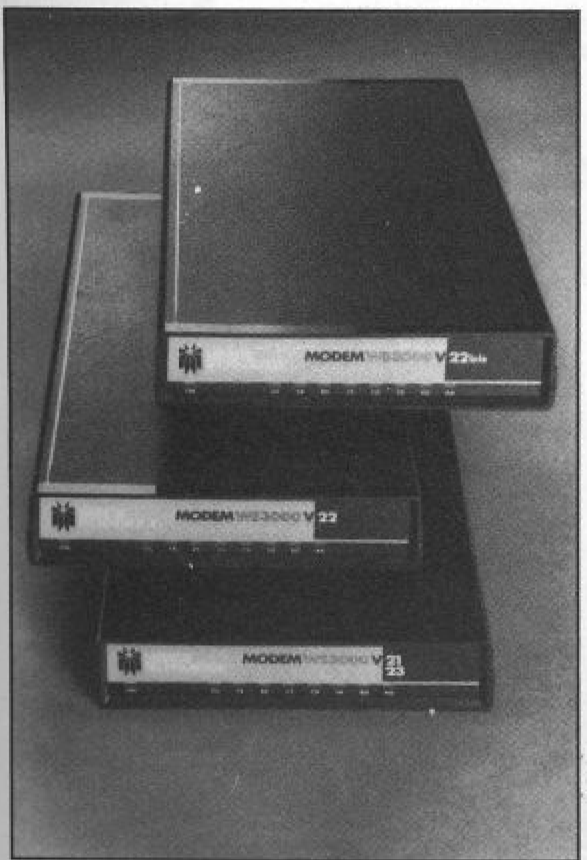
ALL MIXED UP

CONTENTS

COMPETITION

SERIES

REVIEWS



REGULARS



DATA STATEMENTS

Is it a bird? Is it a plane...

The 8th Personal Computer World Show

4-8 SEPTEMBER 1985
OLYMPIA, LONDON
Sponsored by Personal Computer World

THE PERSONAL COMPUTER WORLD Show is the venue for the preview of Beyond's latest title – *Superman*. The official launch date is set for October and the game will be available on the C64 on both cassette and disk.

Beyond's managing director, Bill Delaney, is very confident that the game will be a resounding success. He said: "The game has appeal for everyone – easy to understand with plenty of action and interest for the most avid games fan."

Earn your stripes

IF YOU RATE YOUR PROGRAMMING skills highly then you may be interested in doing some professional work for Tigress Marketing.

The company is often approached by publishers and asked to recommend programmers to make Tigress designs into commercially viable software.

If you specialise in music, adventures, arcade games, strategy, graphics, animation or speech then send details to Tigress Marketing and don't forget to say whether you have any professional experience or not.

Contact Julie Garnett, Tigress Marketing, Suite 12, Hamilton Hse, 66 Upper Richmond Rd, Putney, London SW15 2SQ or phone: 01 871 1136/37.

Quiz kids

THE BCS/COMMODORE SCHOOLS Computer Quiz has been won by Warwick School. The members of the winning team were Greg Michael, Chris Purvis and Tom Duffy. They beat a team from Woodchurch High School and received the first prize of £1600-worth of computer equipment for their school.

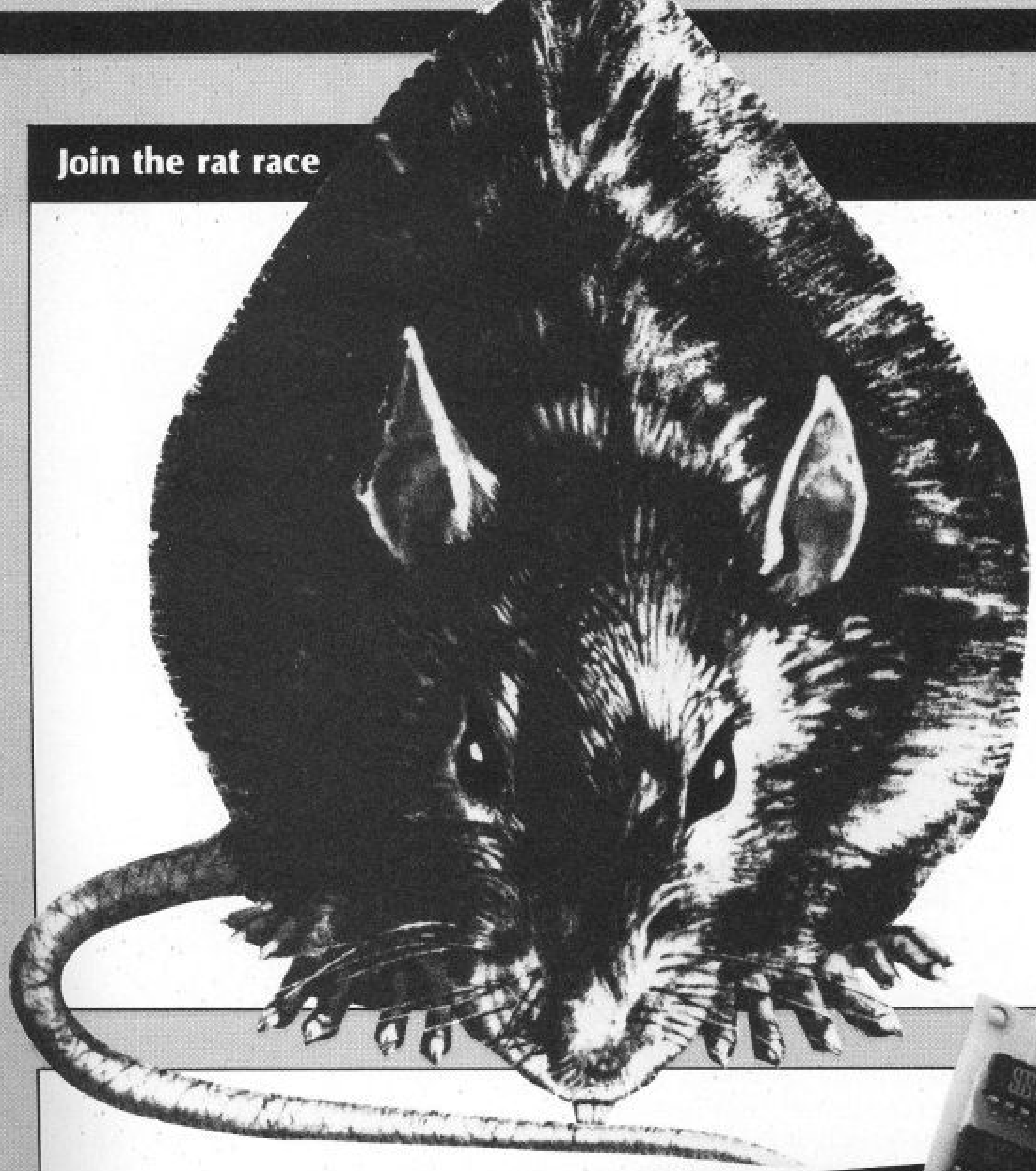
The final was hosted by Michael Rodd – a long standing TV quizmaster. It took place at the University of Reading and Dr Ewan Page, the vice-chancellor, who is also President of the BCS, presented the Momentum Award to Chris Purvis and Dermot Wilson who received the highest scores of the day.

Team members also received individual prizes of a voucher for a free week in Camp Beaumont on a computer course.

Over 400 schools entered and Commodore's Computers in Schools scheme loaned 600 computers to educational establishments for the Quiz. The schools now have the chance to buy the equipment at half price.



Join the rat race



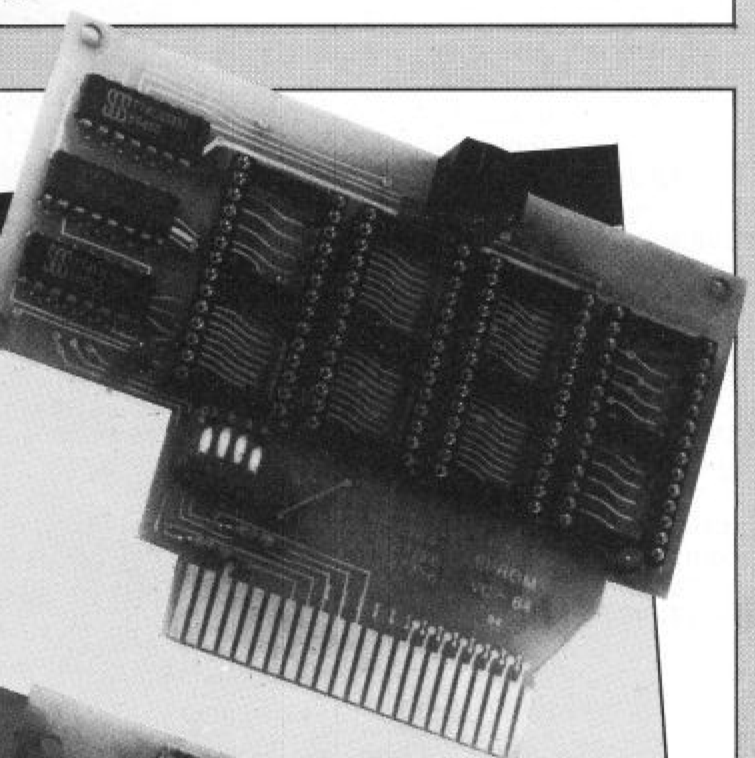
HODDER AND STOUGHTON SOFTware has now brought an aspect of horror to your computer screen with a new adaptation of James Herbert's best selling novel, The Rats.

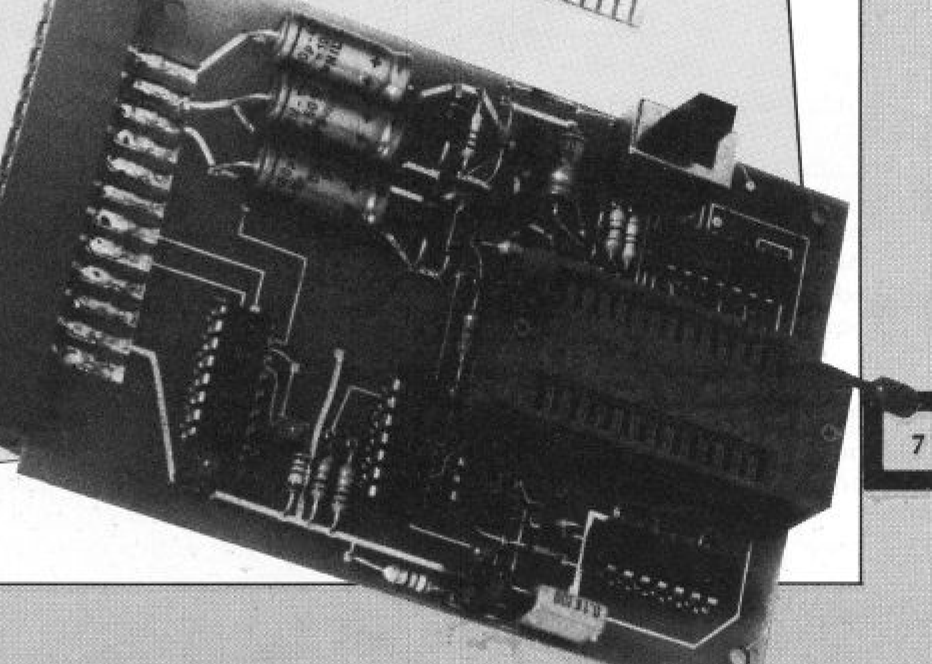
Blood thirsty rats are roaming the streets of London and you are the person unfortunate enough to have been picked to save the world - again. From your control centre you deploy your forces to combat the menace and then you must venture abroad so you can assess the situation first hand. This section of the game leads you through the streets of London, along the banks of canals and into derelict areas.

According to the makers, the game is a combination of strategy and adventure. The adventure section contains animated graphics.

Available now on the C64 it costs £7.95. Contact: Hodder and Stoughton Software, PO Box 702, Dunton Green, Sevenoaks, Kent TN13 2YD.

Burnt into memory





A NEW EPROM BURNER SPECIALLY DEsigned for the C64 and 128, has recently been released by Lightwave Leisure. Lightwave also supplies 8K and 128K mother boards for use with the EPROM Burner.

EPROM modules enable the user to dispense with LOAD times from cassette or disk and are not harmed by magnetic radiation or grease.

It is possible to erase the programs on the EPROM with ultra-violet radiation and it can then be re-programmed several times.

A software operated register makes the EPROM programs available to the computer at three different address sections in 8K sections.

Full instructions and operating and control software are supplied with the product. The EPROM Burner costs £54.95. The 128K mother board is £34.95 and the 8K board, £12.95. For further information contact: Lightwave Leisure, 2 Maldwyn Rd, Liscard, Wirral, Merseyside L44 1AL.

Fast mover

THE QUICKDISC+ FROM EVESHAM Micro is a cartridge based enhancement program for the 1541 disk drive.

Evesham claims many virtues for its new product including: Fast LOAD and SAVE – four to five times faster – fast FORMAT, fast file copy and fast disk backup on unprotected software.

The utility costs £19.95 and is available from Evesham Micro Centre, Bridge St, Evesham, Worcs WR11 4RY.

Showing off

THERE'S A HUGE VARIETY OF MICRO paraphenalia on display at the Personal Computer World Show, this year. The show is at London's Olympia and open to the public Friday 6 to Sunday 8 September.

Exhibitors include many big names from the computing world including Amstrad, Atari, Acorn and of course Commodore. And there will also be all the latest software on show.

The show occupies two major halls at the exhibition centre and is the biggest so far. All aspects of the computer industry are being covered from home to business, industry to education.

A special attraction is Tomorrow's Micro Home which will incorporate all the latest technology in a domestic setting: there's TV, hi-fi, video and computer linked video-disk plus examples of uses for computers in work and leisure.

There's also a special educational section which illustrates the uses of computers in this area.

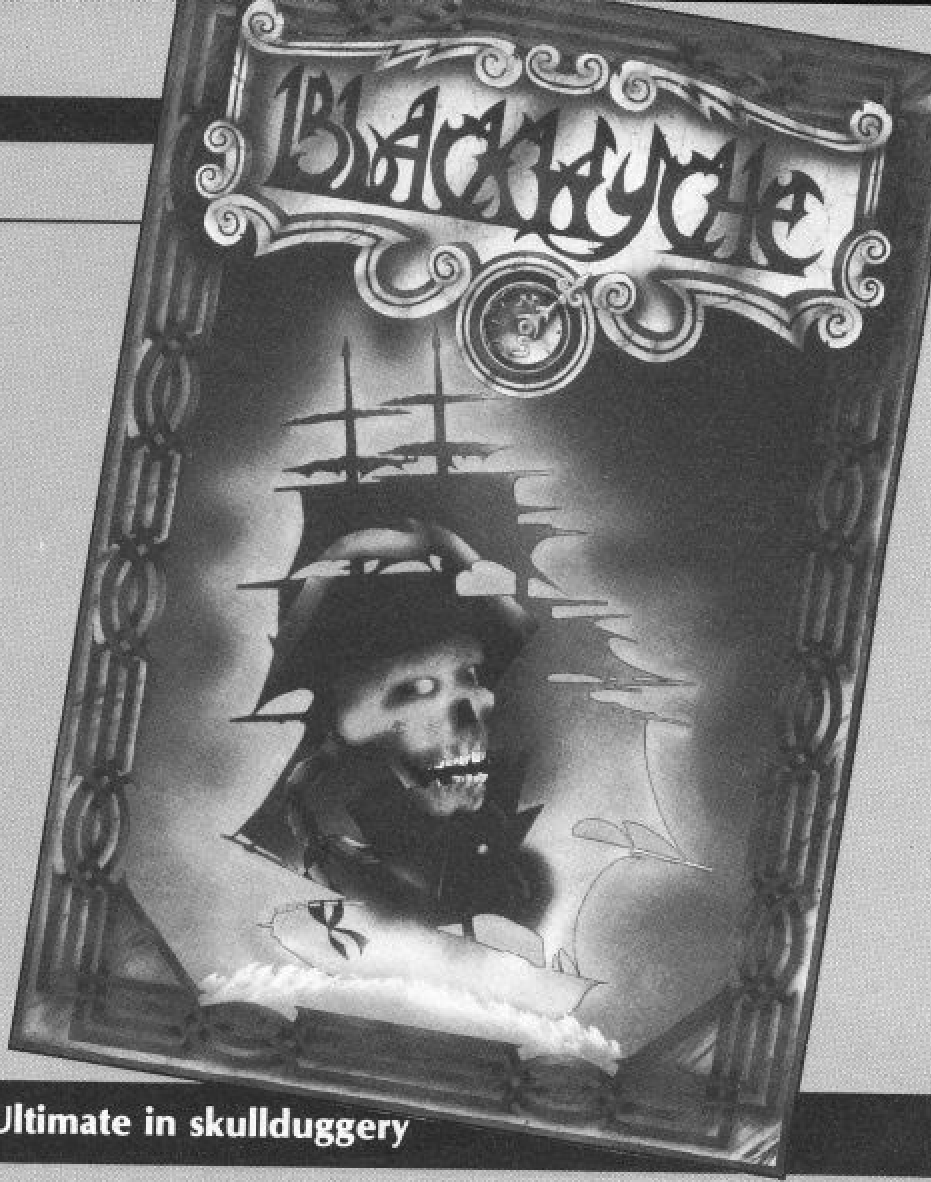
For those who find themselves spoilt for choice when buying a computer there is a team from the NCC Microsystems Centre and members will be giving help and advice plus daily seminars on buying a computer for business use.

Major software exhibitors include Activision, Mirrorsoft, Martech and Mikro-Gen and at the Commodore stand you'll get another chance to have a look at Commodore's new 128.

The show is open from 10am to 7pm except Sunday when it closes at 5pm. Admission is £2 and tickets are available in advance from: PCW Show, 11 Manchester Square, London W1M 5AB.

There will also be an Argus Specialist Publications stand at the Show which will display Argus' wide range of computing publications covering most aspects of the home micro scene.

Come along and see us and have a browse.



The Ultimate in skullduggery

Ultimate has a new game for the C64 entitled Blackwyche.

The arcade game continues the adventures of Sir Arthur Pendragon in an ocean going quest in which he must find the Crystal Skull of Souls on an island shaped like a skull.

The game will retail at £9.95 and Ultimate's address is The Green, Ashby de la Zouch, Leics LE6 5JU.

Eyes down



CRL, A FIRM ALREADY WELL KNOWN for its software, has now moved into the peripherals market with the launch of the Video Digitiser Module for the C64.

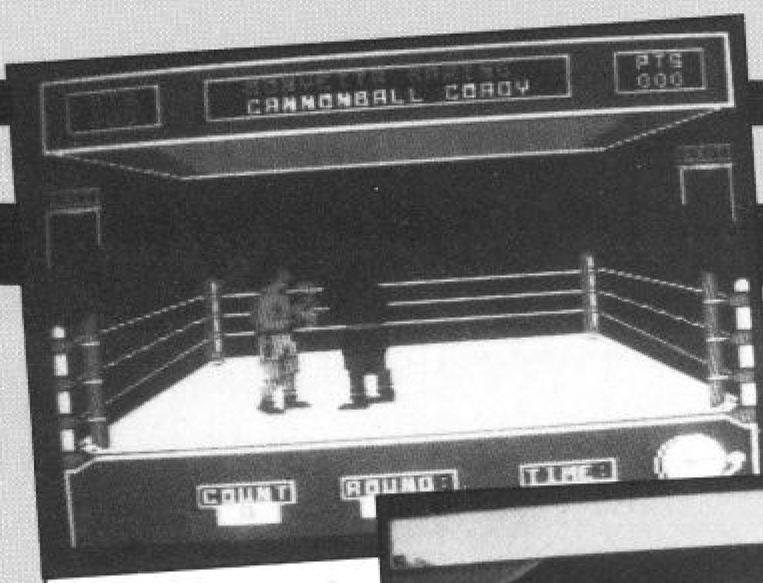
The unit allows video signals to be displayed on the screen, stored to disk and printed out.

The Austrian designed Digitiser is compatible with a normal video camera, video surveillance camera, video recorder, video-out on a TV or the output from a weather satellite receiver which is video compatible.

The product displays an image on the screen and the user can pan, with the aid of the cursor keys, to view the whole picture. Initially in four shades of grey, colours can be incorporated using the function keys. Using a lightpen, sections of the picture can be isolated and used later as user defined graphics.

The Digitiser retails at £149.95 and CRL claims it brings sophistication and ease of use, previously out of reach of the home computer owner.

Contact: CRL, CRL Hse, 9 Kings Yard, Carpenter's Rd, London E15 2HD.



From bikes to boxing



ACTIVISION HAS PRODUCED A COMPREHENSIVE range of games for the autumn varying from a simulation of the Tour de France to the discovery of a tiny being living inside your 64.

The company has signed up world champion boxer, Barry McGuigan, to endorse its new boxing game. Released at the end of August, Activision claims that the game "incorporates his fighting style and unique skills right into the gameplay."

The Great American Cross Country Race is a driving simulation which takes you on a coast to coast trip across the USA. Terrain, time of day, weather reports and distance must all be taken into account so there's plenty to think about. You must also watch your fuel and look out for police speed traps.

Somebody's in My Computer is a totally new idea in games. According to Activision, a little known fact is that there is a tiny person living inside every computer. Activision now has the technology to enable you to see and make contact with this little friend. You can tell him to do things, take care of him or just watch as he goes about his business. And no two are the same.

Get on your bike and take part in the *Tour de France*, the official version of the great French push-bike marathon. The game relies on joystick dexterity to pedal, steer, brake and change gear around the 16 stage circuit. And if you think all sounds too down to earth for you then one of Activision's other offerings may be your cup of tea. *Rescue on Fractulus* is a space pilot simulation which lands you with the difficult task of saving your fellow pilots who have been stranded on the hostile planet, Fractulus.

For utility enthusiasts, Activision is offering the chance to design your own games the easy way with *Gamemaker*. Designed by Garry Kitchen, the makers claim that it can help would-be programmers to create almost any game imaginable.

All the new titles are £9.99 on cassette and £14.99 on disk except Game Maker which is £14.99 and £19.99 respectively.



A similar product, recently released, is Computereyes from Stem Computing in Dundee.

Computereyes is a video acquisition for the C64 and enables images to be captured on screen and then dumped to a printer.

It connects the video source to the 64 via the User I/O Port and a grey scale picture appears on the screen in about six seconds.

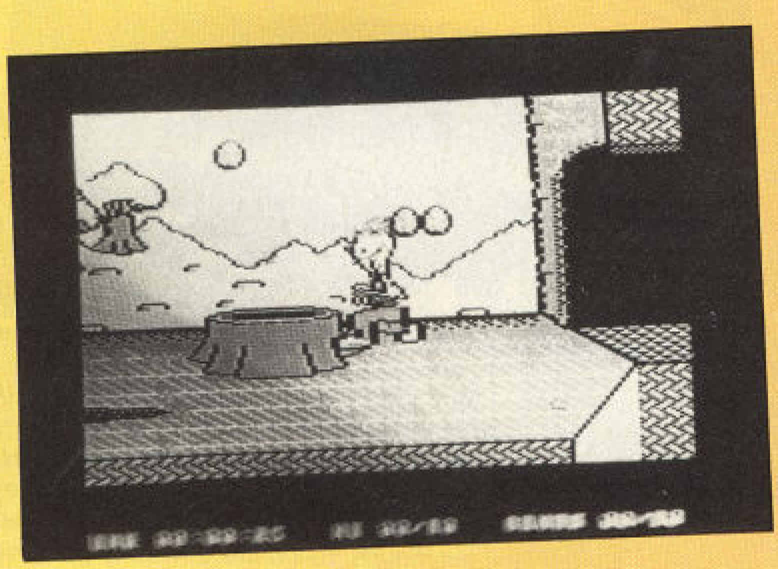
Software is provided which includes machine language image capture routines, a menu-driven executive and image packaging routines.

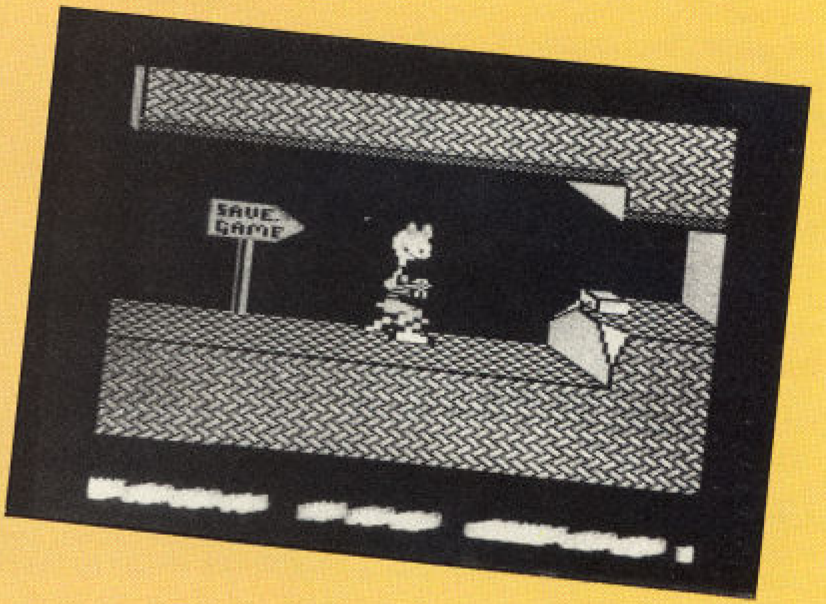
According to Stem Computing, there are numerous uses for the product: pattern recognition, security, spatial measurement and education.

Computereyes costs £139. A demo disk is available for £10 and a source code disk for £40. Stem Computing can be contacted at 3 Blackness Avenue, Dundee DD2 1ER.

William Wobbler is the latest Tony Crowther creation from Wizard Development. Jayne Goin has been delving into its secrets for your enlightenment.

GAME OF THE MONTH





THE QUALITY OF C64 GAMES HAS IMproved in sophistication over the past year, mainly as a result of the American invasion of the British market. Now our own software houses are fighting back.

William Wobbler by Wizard Development's Tony Crowther illustrates this upward curve by exhibiting not only graphical complexity but also a well planned gaming system.

The hero of this new action adventure is an odd looking, slinking character who shuffles across the screen in search of small pieces of a larger puzzle, the solution of which carries a prize worth £1000. The only way to find out what the prize will be is to buy the game and solve the puzzle because the pieces form a picture of the prize.

William's world is a labyrinth of tunnels and passages which are entered by jumping down hollowed out tree-stumps. There are 10 stumps in all laid out across the length of almost as many screens. As William walks his head wobbles up and down and although he can duck under flying frogs and falling rocks his head has a nasty habit of wobbling back up quite quickly so timing is vital.

On entering a hole at the beginning of the game the way is often blocked by a droid which can only be destroyed by the glowing orb hidden in the only droidless hole in the game. The pathway to the orb is punctuated with small pools of water and stepping stones which William must jump if he is to succeed. Touching the water literally makes him fall to pieces and as if that wasn't enough, there are falling rocks to be ducked.

The only way to enter another hole after negotiating your way back from the orb's hiding place is to jump down an exit hole to the lower passageway which is inhabited by flying frogs and snakes which mean instant disintegration for William if he's not careful. At the end of this passage is a rope to climb but players will also see a signpost marked 'Save Game'. Following along in the direction of the sign brings William to a disk drive which allows you to save your position to tape but first you must find the disk which operates the drive. Nothing is easy in this game.

On reaching the surface again, William has to get down another tree-stump without being hit in the back of the neck by a pesky bluebird or by being clobbered by the club carried by his irate mother-in-law.

When William needs any object which he has collected, it appears automatically, demonstrating whether or not the thing you are facing is dangerous or not. Or does this mean the object has to be picked up?

This is the major difficulty because you only get one life per game and a fatal choice means the loss of all your objects and a restart to the game, unless you have found the disk which does not appear until you have ventured well into the maze. There is also an object in one of the caves which will make you drop everything, you are carrying and replaces them where they were originally.

One of the problems to be overcome would make Indiana Jones cringe. A gigantic red ball rolls towards William, totally blocking the tunnel. There is a way to pass this point but I'm not allowed to divulge the secret, however I will say that there is a key object hidden on the other side of the ball.

Some of the tunnels end in doorways which are always locked and frequently booby trapped. The trap could be on the outside or the inside and the keys are hidden elsewhere in the labyrinth. For example, one key hangs from the roof of a cave. To reach it William must have a piece of pipe but its retrieval blocks off the only exit. To escape he must shoot at a target but this implies the ownership of a gun. No gun, no chance.

Scattered liberally through the caves are the clues for the competition and a tally is kept of the number found on a wobbly scrolling scoreboard at the bottom of the screen.

This game makes full use of the 64 and it is rumoured that their are only a handful of bytes left unused in this mega-adventure which will become an essential addition to any gameplayer's library.



A whole new world awaits you and your 64. The world of Prestel, Micronet, Viewfax, Homelink, Citiservice, Compunet (Commodore's own database) etc. This is the world of product reviews, mainframe games, home banking, business information, commodity futures, armchair shopping, real time conversation, electronic notice boards, helpful tips, ICPUG, news, Telex-Link, Mailbox, Chatline, Gallery etc. This is the world of telesoftware, with programmes to download into your 64 and save to disk or tape. Many telesoftware programmes are provided free of charge or at nominal cost.

Dialog with over 600 massive databases on call and almost unlimited information on just about any subject is accessible from your 64, together with BT Gold, Easylink, One to One etc., with electronic mail, business services and telex facilities.

You can link to Bulletin Boards in your area for local news, club news, letters, jokes, poems, helpful tips, etc. It's a whole new experience.

Take the first step. Make a giant leap for your 64. Communicate with Modem House and you'll soon be talking to other '64 users, linking to other micros and swapping files, tips, programmes or just chatting.

Modem House are the largest suppliers of Commodore communication equipment. We also supply complete systems for the Pet 2000, 3000, 4000 and 8000 series.

MODEM HOUSE

Iolanthe Drive, Exeter, Devon EX4 9EA. Tel: (0392) 69295

The most popular 64 micropack in use today, will link you to Prestel, Micronet 800, Viewfax 258, Homelink, Farmlink, Citiservice, Compunet etc. From Modem House, the complete system is on offer at the incredible price of £99.95 inc. VAT, p & p.

Optional terminal software to link you to BT Gold, Easylink, One to One, Dialog and Bulletin Boards is available at £29.95 inc. VAT, p & p.

Complete Commodore 64 micropack

special price **£99.95** inc. VAT p & p
for a limited period only.

Special, special offer, order your 64 micropack now and get a **FREE** quarter's subscription to Micronet 800 and Viewfax 258 (Prestel Microcomputing).

Modem House, Iolanthe Drive, Exeter, Devon EX4 9EA. Tel: (0392) 69295

Please send me .... Commodore 64 Micropack(s) £99.95 each
.... Optional Terminal Software £29.95

I enclose £.................. /Bill my Access No. ....

Name ....

Address ....

....

....

Telephone No. ....

Please allow 28 day delivery

This offer applies only while stocks last

YC 10/85

IN·ARCADIA

This month Bounty Bob comes out on top and Rock'N'Bolt rocks into Arcadia, as Flippo gives you the low down.

GREETINGS, ACTION FANS! WE BOUNCE back this month with some fabulous tips and tricks for your own arcade games programs, plus a gander at some of the latest offerings to hit the shops, and strategies for some new games.

Poke in the eye dept.

Here are some interesting POKEs for you to use in your games programs, to clobber the chances of any unscrupulous person, i.e. me, from cracking your game and stealing it.

POKE	Function
775,191	disables LIST
775,167	enables LIST
819,246	disables SAVE
819,245	enables SAVE
816,157	disables LOAD
816,165	enables LOAD
808,239	disables RUN/STOP
808,237	enables RUN/STOP
792,193	disables RESTORE
792,71	enables RESTORE
646,x	changes character colour, where x=any No. from 0 - 7
22,35	removes all the lines No.s in a Basic program.
22,25	puts them all back

(You could just as easily put these into an Assembly Language program as you could in a Basic one.)

Transatlantic Preview

Track and Field (Konami/Atari) is the best sports game I've played since Daley Thompson first threw a javelin. This is a conversion from the real arcade machine which started the whole sports game fetish in the first place. You can play against the computer or a real opponent, with a split screen depicting each player's progress, a lot like Pitstop II.

I think it's the best, but that's just my opinion. Try it yourself, when it comes out over here!

Bob's yer uncle

I've been playing *Bounty Bob Strikes Back* (Big Five/US Gold) lately, and I have achieved a certain competence; that is to say I breached the elusive 30,000 score, which to my mind constitutes a reasonable score on most games (good rule of thumb, that). My current hi-score is 46,880, and I got to Bob's Playroom as well (Level 4, I think!).-

I like this game for several reasons. It plays well, the music is entertaining and not in the least irritating, and it has the most amusing hi-score table I've ever seen. The sort routine to enable the little birds to pick up the letters and place them in the table is an awesome piece of programming, and that is before you even get to the game. A lot of games claim "arcade quality", but *Bounty Bob* has it. I'd happily push my 10 pences into this one.

Let's do the time warp again II

I keep mentioning the *Rocky Horror Show*, as I've had the Spectrum (arrrrghhhh! Pass the Crucifix!) version for some time. All my observations on the game had until now been based on this altogether inferior version. I can report with pleasure that the new C64 version is better! It's more difficult to complete, the music is what a friend of mine would call fazz (good), and the sound effects are more than appropriate. BUT (and as you can see, that's a big but) the sprites let it all down a bit. I've seen multicoloured sprites that would knock yer eyes off, but these are chunky to say the least, and more multicoloured lo-res to say the most. Otherwise a super game and certainly well worth a bash.

Hot tips

Bounty Bob: The way the joystick controls Bob can be very precise. To jump onto a close platform push the fire button and just nudge the joystick in the desired direction on his way down. Ignore the mutants and only kill them if they're getting in your way. Highest scoring settings: difficulty on medium, bonus at 10,000, and lives four.

Strangeloop: Tricky this one. Best advice I can give you is to blast all the swarf in every room before you move on, to make sure it can't multiply again.

Rock'N'Bolt: Make good use of the Practise Mode (unlimited lives) to suss the levels out before you begin a game.

Better luck next time dept.

Pastfinder (Activision) promised to be a really good game. The graphics ain't bad and the sound is fairly good...but it's so flippin' boring! Who cares whether you make it to the top of the screen or not? Activision should really make the senarios more entertaining, as I really couldn't bear to play this game for more than 10 minutes before I reset and loaded Bounty Bob again. Try it out in the shop before you buy it, and see what you think.

Breakdance (EPYX/CBS) should be very good, but I couldn't shake the feeling of playing a sort of breakdancing space invaders, and paying a lot of spons for the privilege! Technically a very good program, but largely uninteresting and, I suspect, just a piece of bandwagon hopping. Epyx doesn't need to do this. It's a good enough company to make much more original games, without this kind of cheap shot.

Ooops! my mistake dept.

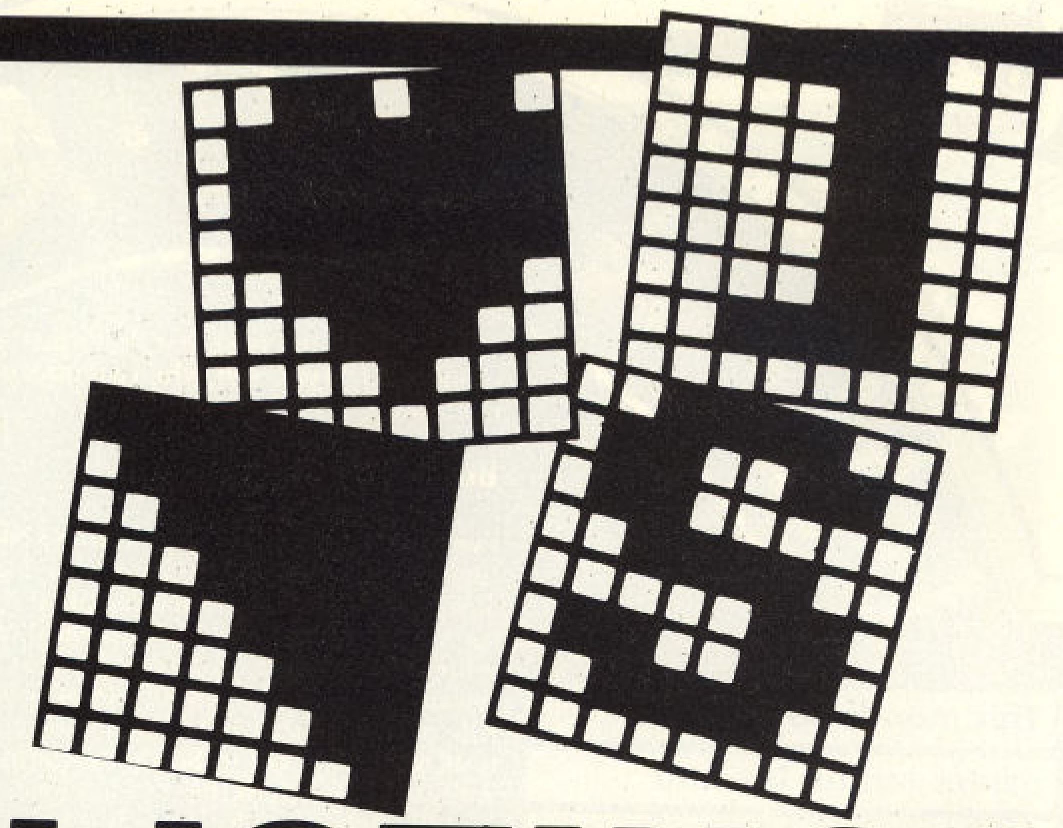
My profuse apologies to English Software. Last month I featured Henry's House in my Transatlantic Preview spot. I was misinformed as they are actually a bunch of Mancunian lads, revealing a trap I for one always fall into; just because a piece of software has superb state-of-the-art graphics doesn't mean that the said piece of software comes from across the Atlantic. There is a great deal of World Leading programming talent on this little island (lots that we don't know about yet too!) and I promise I won't forget that from now on. (Take 100 lines, Flippo - Ed.)

I reiterate my appraisal of last month: Henry's House is a very good game. Support your local programmers.

Mailroom Special Message

Send in your hints, tips, cheats, bugs and hi-scores to Flippo, c/o Your Commodore, Argus Specialist Publications No. 1 Golden Square, London W1R 3AB, and see your name in print.

Listings will be much easier to enter with our new system.



LISTINGS

COMMODORE LISTINGS ARE RATHER well known for the horrible little black blobs that always abound. Unfortunately the graphics characters which are used to represent graphic and control characters do not reproduce very well and they are also difficult to find on the Commodore keyboard.

For this reason Your Commodore started to precede any control characters with a REM statement on the previous line that explained exactly what the black blobs were meant to be. Unfortunately the graphics characters were not documented and these still cause some confusion. For this reason we are starting to use a new method for marking the control and graphic characters in our listings.

In future all control and graphics commands will be replaced by mnemonic within square brackets. This mnemonic is not typed out as printed in the magazine but rather the corresponding key or keys on the keyboard are pressed. For example [RIGHT] means press the cursor right key, you do not type in [RIGHT]. All of the keywords, what keys to press and how they are shown on the screen are shown below.

Any character that is accessed by pressing shift and letter will be printed as [s LETTER]

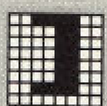
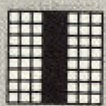
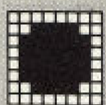
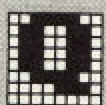
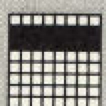
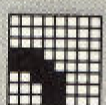
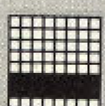
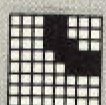
[s A]	shift and A
[s C]	shift & C

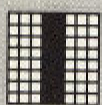
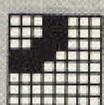
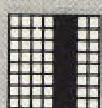
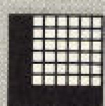
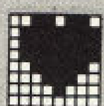
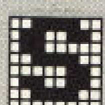
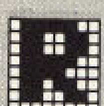
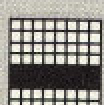
Any character that is accessed by pressing the Commodore key and a letter will be printed as [c LETTER]

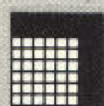
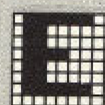
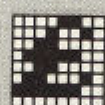
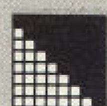
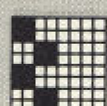
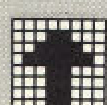
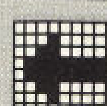
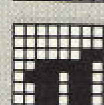
[c A]	Commodore & A
[c C]	Commodore & C
[c 1]	Commodore & 1

Any control key will be printed out as a number. For example [001]. Control codes are accessed by pressing the CTRL and a letter at the same time [001] is CTRL & A, 002 is CTRL & B etc. See the manual for more information about control codes.

[001]	CTRL & A
[026]	CTRL & Z

Mnemonic	Symbol	what to press
[RIGHT]		left/right
[LEFT]		shift left/right
[UP]		Shift & up /down
[DOWN]		up/down
[F1]		f1
[F2]		shift & f1
[F3]		f3
[F4]		shift & f3

Mnemonic	Symbol	what to press
[F5]		f5
[F6]		shift & f5
[F7]		f7
[F8]		shift & f7
[CLEAR]		shift & CLR /HOME
[HOME]		CLR/HOME
[RVSON]		CTRL & 9
[RVSOFF]		CTRL & 0

Mnemonic	Symbol	what to press
[BLACK]		CTRL & 1
[WHITE]		CTRL & 2
[RED]		CTRL & 3
[CYAN]		CTRL & 4
[PURPLE]		CTRL & 5
[GREEN]		CTRL & 6
[BLUE]		CTRL & 7
[YELLOW]		CTRL & 8

Teacher's Pet

Margaret Webb continues her look at maths with this round up of programs.

Maths Part Two

LAST MONTH WE LOOKED AT HOW THE C64 along with some software, can help the pre-school child get to grips with basic number learning.

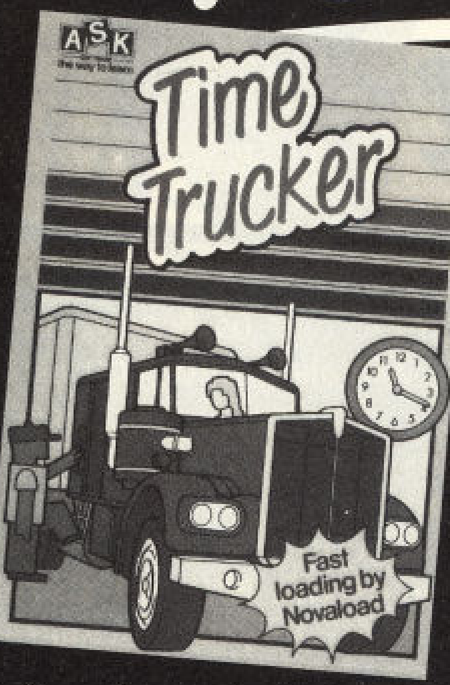
Once this, the most important hurdle, is over and the child is at school, the computer must take a secondary role in his education. This is not because there is no suitable software available – quite the contrary – but because the child should be doing sufficient at school without extra work being thrust upon him at home. However, if your child is slipping behind with his work or if he shows a real enthusiasm and wants to do extra, there are numerous programs to look at.

Let's start by looking at some programs which re-cap the four basics of arithmetic – addition, subtraction, multipliction and division. There are numerous programs which deal with this aspect of number work, it is also the easiest to get wrong. Some programs are straightforward teach and test and can be very dry and boring. These can do more harm than good. Others use the capabilities of the 64 to create eye-catching graphics which grab the child's attention and hold it so the child is learning without really being aware of it. At any age, more learning is done in an atmosphere of fun than by having it force fed. Several Publishers have recognised this and in consequence have produced some fun to play games which also teach and/or test.

Shards Software has a lovely program in the *Fun to Learn* series. *Monster Maths* is a collection of programs which cover size estimation, mental arithmetic, multiplication tables and logic. Each game has a different format but is written to give maximum fun whilst still helping the user to learn. In one game the player feeds in a friend's name and then tries to answer three questions correctly in a specified time. If the player is successful his friend is eaten by a monster, if not he himself is eaten.

Another part of the tape covers multiplication tables – one of the bugbears of education. I believe they are an essential part of mathematical training. Should your child's school be one where they are not taught there are some packages on the market to help.

Collins produces *Know Your Tables*. This is a cassette and booklet set which works steadily through tables up to 10. The booklet has lots of related exercises



and shows the user the patterns formed by the numbers.

McGraw-Hill has just released a collection called *Ladders to Learning*. Included in this series are cassettes dealing with addition, subtraction and problem solving. There are also titles dealing with multiplication and division. The child is led through the maze of learning tables by a cartoon character, Henry Hilo, and his friend a number machine. Once multiplication is mastered division can be tackled, and this time Henry shows that if the child can do subtraction and multiplication he can divide.

My overall favourite for learning and using basic number rules is *Number Tumblers* by Fisher-Price. This is a cartridge based game in which you have to bounce a figure around a grid using the numbers and operators to make a series of totals. The game can be played by children and adults alike as the parameters can be set to a very simple or extremely difficult level. The game can be fast and furious and calls for quick thinking especially when playing against the computer.

These basic four rules cover many other topics: telling the time, using money, fractions, decimals and graph work. Collins and Good Housekeeping both have packages dealing with telling the time. *Mr T Tells the Time* uses the computers graphics to create pictures to teach the position of the numbers on the clock face and the rudiments of telling the time. Collins' *What's the Time* uses a cassette and booklet approach to learning about the quarter hours, half hours and hours. It uses both analogue and digital clocks in its examples.

For a completely different approach look at A.S.K.'s *Time Trucker*. This consists of three games in which the player guides a truck around a map collecting fruit and vegetables and taking it to market. In the first game, every time the driver stops to collect produce he has to clock in; however the clock on the screen is an analogue clock and the time has to be entered in a digital form. All good fun and very educational.

Fractions, decimals and graph work are all covered by the *Ladders to Learning* series. There is also a cassette which explains algebra. They follow the same formula as the previously mentioned *Ladders to Learning* series. Henri Hilo explains simply and logically each subject making full use of the graphics capabilities of the machine. Test questions are set to see if the child is understanding everything.

This sort of program should not be used as matter of course at home. If your child is having problems at school or wants to do extra work, see the class teacher involved before deciding to purchase anything and then try to have a good look at the programs available in your local shops. Whatever you do, don't force your child. You may put them off mathematics and thus defeat the object of the exercise. Numbers can be magic and your Commodore can help you show your child the fun.

Other Programs to Look Out For

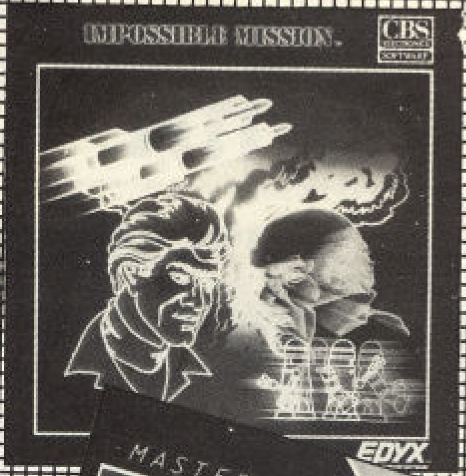
Sprites/Snowmen by Commodore – addition and multiplication games.
Number Puzzler by Commodore – five games dealing with the four rules.
Mathematics by Longmans – '0' level revision package.
Numbers at Work by Collins Brainpower – maths education for adults. Covers fractions, decimals, square roots and V.A.T. and P.A.Y.E. problems.

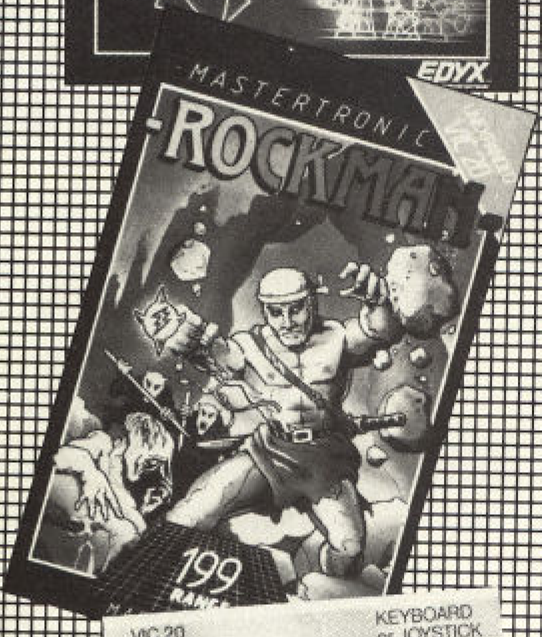
Addresses

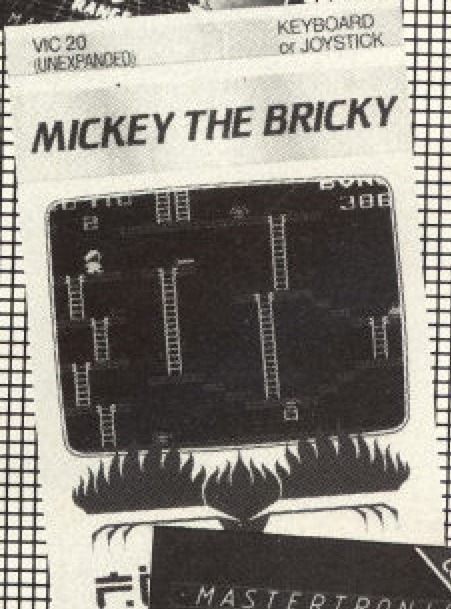
Commodore UK, 1 Hunters Way, Corby, Northants
Collins, 8 Grafton St, London W1E 7JZ
A.S.K., 68 Upper Richmond Rd, London SW1 2RP
Longmans, Fourth Ave, Harlow, Essex CM19 5AA
Good Housekeeping (Eury Software), 72 Broadwick St, London W1V 2BP
McGraw-Hill, Shoppenhangers Rd, Maidenhead, Berks SL6 2QL.

TOP 20 Software

Compiled by Gallup











COMMODORE 64

Title	Publisher
1 The Way of the Exploding Fist	Melbourne House
2 Elite	Firebird
3 Soft Aid	Various
4 Pitstop II	US Gold
5 Dambusters	US Gold
6 A View to a Kill	Domark
7 Kikstart	Mastertronic
8 International Basketball	Commodore
9 International Tennis	Commodore
10 Dropzone	US Gold
11 Shadowfire	Beyond
12 Impossible Mission	US Gold
13 Cauldron	Palace
14 Entombed	Ultimate
15 Graham Gooch's Test Cricket	Audiogenic
16 Action Biker	Mastertronic
17 Theatre Europe	PSS
18 Rocky Horror Show	CRL
19 Jet Set Willy II	Software Projects
20 Bruce Lee	US Gold

Retail sales for week ending July 19, 1985.

VIC 20 Top Ten

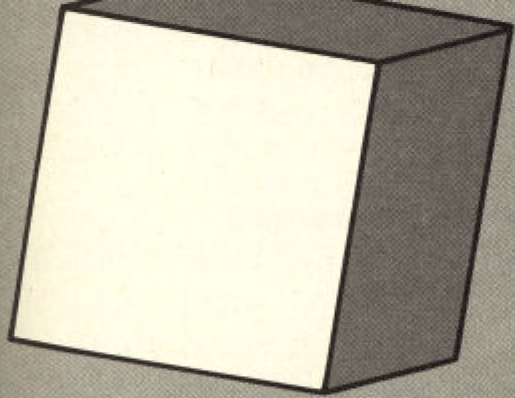
Title	Publisher
1 King Tut	Mastertronic
2 RIP	Mastertronic
3 Rockman	Mastertronic
4 Football Manager	Addictive
5 Mickey the Brickey	Firebird
6 Vegas Jackpot	Mastertronic
7 Bullet	Mastertronic
8 Snakebite	Firebird
9 Hunchback	Ocean
10 Psycho Shopper	Mastertronic

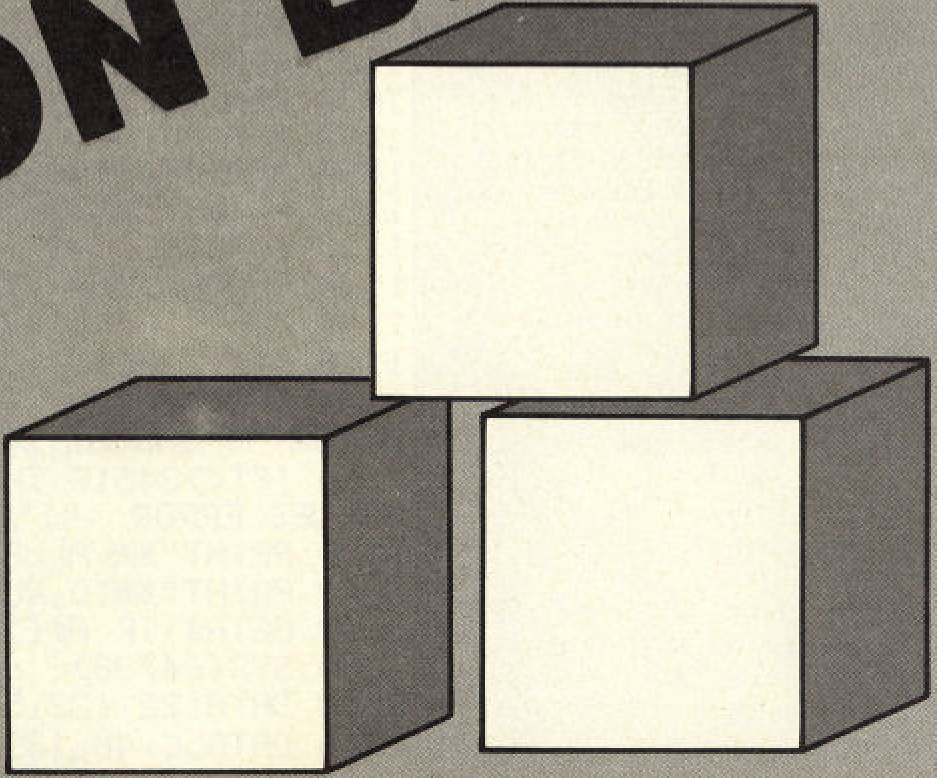
Retail sales for week ending July 19, 1985.

Compiled by Gallup for the industry's weekly trade magazine, Computer and Software Retailing. For details contact John Ross, Computer and Software Retailing, 222 Regent Street, London W1R 3AB. 01-434 2131.

BUILDING ON BASIC

In the first of a new series, Nick Hampshire brings you the first pieces of a bigger, better Basic.





HOW MANY TIMES HAVE YOU thought how nice it would be to have a Basic command to perform a particular function? The kind which would normally require a convoluted piece of Basic code, or a short machine code routine called by a SYS command. Adding extra commands to Basic on the C64 is not difficult and can make writing Basic programs a lot quicker and easier.

There are of course a considerable number of commercially produced extended Basic packages. But, being able to add and create your own commands is much more fun and allows the commands to be tailored to your particular needs.

This article is the first of a series which will include the code for adding a whole range of very useful commands to your Basic interpreter. This article includes the code for the all-important control wedges which allow the new Basic commands to work. It is essential that these routines are in memory before any attempt is made to add command routines.

The following routines are the start of the Basic extension code. These are the main control routines which patch the extra commands into the C64's Basic. They should be used in the order they appear in this article and the accompanying listings.

Initialisation

This section of code contains the initialisation routines and the table of added commands and their vectors. The commands are initialised by calling the cold start ($FFC2 - 64738) which simulates a standard power-up. The routines cannot be used with a cartridge in place as they take up the same locations and are designed to be capable of being placed on a cartridge ROM if required.

The routine labelled 'COLD' is the power-up routine and the routine labelled 'WRST' is the NMI routine. The NMI routine makes sure that the function key and lister wedge routines are not disabled. Note that the table of added commands and their addresses can be changed and added to by the user. The commands given in the table refer to commands which will be added to this package later in the series. Any reader assembling this portion of the code will get a 'Label not present' error from the CADDR table unless a dummy routine is set up for every added Basic command which has not yet been implented. This routine simply takes the following form:

```
XXXXXX JMP $AF08
```

where XXXXXX is the label. Thus, since the first two commands are included at the end of this article a total of 30 dummy routines are required at this stage, which can be removed as the appropriate commands are added. Such a set of dummy routines is shown in the file. LIB DUMMY at the end of the listings.

Crunch to Tokens

This routine is wedged into the Crunch-token link at locations $0304-$0305 (772-773). Crunch to tokens will take the input line and convert all command words to one (normal) or two (Extended) byte token values. This does exactly the same as the original Basic version except that the extended keyword table is checked before the normal Basic table.

Crunch to tokens is done directly after the warm start routine encounters a carriage return. It makes no difference whether the command is direct mode or for entering or deleting a line in memory.

Tokens to Text

This routine is wedged into the Print-token link at locations $0306-$0307 (774-775). Tokens to text is used in the list command only to convert any token value (greater than 127 for Basic or preceded by $EE - 238 for Extended Basic) back into the copmmand word and print it to the output device.

Execute Statement

This routine is wedged into the Start new Basic code link at locations $0308-$0309 (776-777). This is the control part of the main Basic interpreter loop and takes a token value and executes the routine via the vector table in the initialisation.

There is a special case routine for PRINT which uses the same token as in normal Basic but the routine has been re-written to allow the CTL command.

Execute Arithmetic

This routine is wedged into the Arithmetic link at locations $030A-030B (778-

779). This routine is called by Evaluate expression and transfers control to one of the four arithmetic routines included in this package. If the Extended Basic command is not one of the four arithmetic routines, 'syntax error' is output.

Function keys

This routine is wedged into the keyboard table set-up vector at locations $028F-$0290 (655-656). The routine checks if the computer is in direct or program mode. If in direct, the normal routine is executed, if in program mode, the quotes flag is checked and if set, the normal routine is executed.

The current key pressed is checked for one of the four function keys and the shift key. If it is a function key, the text for that key is read from behind the Basic ROM and put into the keyboard buffer until all eight characters or a zero byte terminator is found. If not a function key, the normal routine is executed.

Program Lister

This routine is wedged into the INPUT vector at locations $0324-$0325 (804-805). It exactly simulates the normal input routine. First the input device is checked for keyboard. If not found, the normal routine is executed. Direct mode is then checked for and if not found, the normal routine is again executed.

The next part of the routine is copied directly from the kernal routine except that the cursor down key is checked for and, if found, then the cursor position is checked. If the cursor is not on the bottom line of the screen, the cursor down character is printed. If the cursor is on the bottom line, instead of printing cursor down, the next line number is found and that line listed (any output device).

NOTE: there is no check for quotes so if you're entering a line on the bottom line of the screen, the line will be wiped out and a line listed if you press the cursor down key even from within quotes.

When the last line of the program is listed, the cursor will remain at the end of the line, cursor down again will produce the message:

***********END OF PROGRAM********

after this, the program will start listing from the beginning again.

Print

The purpose of this routine is to PRINT characters to the open CMD output channel (usually value three – screen). This version of PRINT does exactly the

Basic Loader

```
1000 REM ******************************
1010 REM *                            *
1020 REM * BASIC LOADER FOR EXTENDED *
1030 REM *       BASIC COMMAND        *
1040 REM *          PACKAGE           *
1050 REM *                            *
1060 REM * COPYRIGHT 1985             *
1070 REM * NICK HAMPSHIRE             *
1080 REM *                            *
1090 REM ******************************
1100 PRINT"{CLR}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}      ENTERING EXTENDED BASIC"
1110 I=32768:T=0
1120 READA:IFA=-1THEN1150
1130 POKEI,A:T=T+A
1140 I=I+1:GOTO1120
1150 IFT<>203934 THENPRINT"{DOWN}{DOWN}CHECKSUM
ERROR:"T"SHOULD BE 203934"
1160 IFI<>34518 THEN PRINT"{DOWN}{DOWN}NUMBER OF
VALUES ERROR :"I"SHOULD BE 34518":END
1170 PRINT"{DOWN}{DOWN}VALUES ENTERED CORRECTLY"
1180 PRINT"{DOWN}{DOWN}TO RUN PRESS ANY KEY"
1190 GETA$:IF A$<>""THEN1200:GOTO1190
1200 SYS(64738)
2000 DATA122,128,57,128,195,194,205
2010 DATA56,48,139,227,131,164,201
2020 DATA129,158,130,247,130,59,131
2030 DATA76,72,178,0,49,234,68
2040 DATA128,71,254,74,243,145,242
2050 DATA14,242,80,242,51,243,241
2060 DATA131,202,241,237,246,62,241
2070 DATA47,243,68,128,165,244,237
2080 DATA245,32,188,246,32,225,255
2090 DATA240,3,76,114,254,32,163
2100 DATA253,32,24,229,32,93,128
2110 DATA32,204,255,169,0,133,19
2120 DATA32,122,166,88,162,128,76
2130 DATA136,227,162,21,160,128,134
2140 DATA195,132,196,160,35,177,195
2150 DATA153,16,3,136,16,248,169
2160 DATA118,160,131,141,143,2,140
2170 DATA144,2,96,142,22,208,32
2180 DATA163,253,32,80,253,32,91
2190 DATA255,32,93,128,88,32,229
2200 DATA128,32,191,227,169,128,133
2210 DATA52,133,54,133,56,169,0
2220 DATA133,51,133,53,133,55,169
2230 DATA172,160,128,32,45,228,162
2240 DATA251,154,208,172,147,13,32
2250 DATA32,32,32,42,42,42,42
2260 DATA32,69,88,84,69,78,68
2270 DATA69,68,32,54,52,32,66
2280 DATA65,83,73,67,32,86,48
2290 DATA49,32,42,42,42,42,13
2300 DATA13,32,54,52,75,32,82
```

Basic Loader

```
2310 DATA65,77,32,83,89,83,84
2320 DATA69,77,32,32,0,162,11
2330 DATA189,9,128,157,0,3,202
2340 DATA16,247,96,82,85,206,67
2350 DATA84,204,65,80,80,69,78
2360 DATA196,65,85,84,207,67,65
2370 DATA84,65,76,79,199,67,72
2380 DATA65,78,71,197,67,72,65
2390 DATA73,206,67,82,85,78,67
2400 DATA200,68,69,76,69,84,197
2410 DATA68,73,83,203,68,79,75
2420 DATA197,68,85,77,208,69,88
2430 DATA69,195,70,73,78,196,71
2440 DATA69,212,75,69,217,77,65
2450 DATA212,77,69,82,71,197,79
2460 DATA76,196,80,79,208,80,85
2470 DATA212,82,69,78,85,77,66
2480 DATA69,210,82,69,80,69,65
2490 DATA212,83,79,82,212,84,82
2500 DATA65,67,69,79,206,84,82
2510 DATA65,67,69,79,70,198,84
2520 DATA89,80,197,85,78,84,73
2530 DATA204,68,69,69,203,72,73
2540 DATA77,69,205,76,79,77,69
2550 DATA205,86,65,82,80,84,210
2560 DATA0,116,134,138,133,119,134
2570 DATA122,134,125,134,128,134,131
2580 DATA134,134,134,137,134,140,134
2590 DATA143,134,146,134,149,134,152
2600 DATA134,155,134,158,134,161,134
2610 DATA164,134,167,134,170,134,173
2620 DATA134,176,134,179,134,182,134
2630 DATA185,134,188,134,191,134,194
2640 DATA134,197,134,200,134,203,134
2650 DATA206,134,166,122,160,4,132
2660 DATA15,189,0,2,16,7,201
2670 DATA255,240,43,232,208,244,201
2680 DATA32,240,36,133,8,201,34
2690 DATA240,71,36,15,112,26,201
2700 DATA63,208,4,169,153,208,18
2710 DATA201,48,144,4,201,60,144
2720 DATA10,76,70,130,169,238,44
2730 DATA5,11,164,113,232,200,153
2740 DATA251,1,201,238,240,49,185
2750 DATA251,1,240,34,56,233,58
2760 DATA240,4,201,73,208,2,133
2770 DATA15,56,233,85,208,174,133
2780 DATA8,189,0,2,240,219,197
2790 DATA8,240,215,200,153,251,1
2800 DATA232,208,240,153,253,1,198
2810 DATA123,169,255,133,122,96,165
2820 DATA11,200,153,251,1,76,207
2830 DATA129,132,113,160,255,134,122
2840 DATA202,169,1,133,11,200,232
2850 DATA189,0,2,56,249,241,128
```

same as the Basic PRINT except that a check has been made for the CTL command to be included. The syntax for this modified command is exactly the same as that of the Basic PRINT command.

The First Extended Basic Command

CTL

Abbreviated entry: C(shift)T
Affected Basic abbreviation: none
Token: Hex $EE,$02 Decimal 238,2
Modes: Direct, program, and in PRINT statements
Purpose: To replace cursor and colour characters, screen and border pokes, thus improving the ability to position the cursor anywhere on the screen. If the value is not specified, the current value is used.
Syntax: CTL ([x][,[y][,[cc][,[sc][,[bc][,[cls]]]]])
Where x is the column position of the cursor (0-39), y is the row position of the cursor (0-24), cc is the cursor colour, sc the screen colour, bc is the border colour (0-15), and cls is a flag for clearing the screen (0 - no, 2 - yes).
Erros: Syntax error – if the syntax is not as above.
Illegal quantity – if the values are out of range.
Use: CTL is a powerful screen handling routine. Cursor, screen, and border colours can be set with a number (0-15), and the cursor can be positioned anywhere on the screen by entering the x position (0-39). There is also a screen clear flag that, if set to '1', will clear the screen before positioning the cursor. To make it easier to describe, here are a few examples and what they do.
CTL (20) positions cursor at middle of current line
CTL (0,0) moves cursor to 0,0 (home position)
CTL (,,1) sets cursor colour to white
CTL (,,,0) sets screen colour to black
CTL (,,,,0) sets border colour to black
CTL (,,,,,1) clears screen leaving cursor at current position
CTL (20,12,5,0,11,1) clears screen (1), sets screen to black (0), border to medium grey (11), cursor colour to green (5), and cursor position to column 20, row 12.
To print something at a specified location on the screen:
PRINT CTL (x,y)"text"CTL(x1,y1)"more text".....
Routine entry point: $88AB
Routine operation: The current settings of the five parameters are read and the screen clear flag is set to 0. The open brackets character is scanned past and each of the six values is read if present, checking to see if there is a closing bracket. When the closing bracket is found, the screen is cleared if the flag is set to 1, and the other values are stored in their own locations.

```
2860 DATA240,245,201,128,240,156,166
2870 DATA122,230,11,200,185,240,128
2880 DATA16,250,185,241,128,208,228
2890 DATA160,0,132,11,136,166,122
2900 DATA202,200,232,189,0,2,56
2910 DATA249,158,160,240,245,201,128
2920 DATA208,3,76,255,129,166,122
2930 DATA230,11,200,185,157,160,16
2940 DATA250,185,158,160,208,225,189
2950 DATA0,2,76,1,130,48,3
2960 DATA76,243,166,201,255,240,249
2970 DATA36,15,48,245,201,238,240
2980 DATA5,32,217,130,48,3,32
2990 DATA186,130,76,239,166,200,177
3000 DATA95,170,132,73,160,255,202
3010 DATA240,8,200,185,241,128,16
3020 DATA250,48,245,200,185,241,128
3030 DATA48,5,32,210,255,208,245
3040 DATA96,56,233,127,170,132,73
3050 DATA160,255,202,240,8,200,185
3060 DATA158,160,16,250,48,245,200
3070 DATA185,158,160,48,230,32,210
3080 DATA255,208,245,32,115,0,201
3090 DATA238,240,10,201,153,240,38
3100 DATA32,121,0,76,231,167,32
3110 DATA14,131,76,174,167,230,122
3120 DATA208,2,230,123,160,0,177
3130 DATA122,56,233,1,10,168,185
3140 DATA138,129,72,185,137,129,72
3150 DATA76,115,0,32,46,131,76
3160 DATA174,167,173,58,131,72,173
3170 DATA57,131,72,76,115,0,235
3180 DATA132,169,0,133,13,32,115
3190 DATA0,201,238,240,6,32,121
3200 DATA0,76,141,174,230,122,208
3210 DATA2,230,123,160,0,177,122
3220 DATA201,29,176,3,76,8,175
3230 DATA133,36,169,173,72,169,140
3240 DATA72,198,36,165,36,10,170
3250 DATA189,138,129,72,189,137,129
3260 DATA72,76,115,0,165,157,240
3270 DATA16,169,1,36,212,208,10
3280 DATA165,203,201,3,144,4,201
3290 DATA7,144,3,76,72,235,197
3300 DATA197,240,249,169,0,133,252
3310 DATA133,251,169,1,44,141,2
3320 DATA240,4,169,32,133,251,169
3330 DATA191,133,252,169,192,24,101
3340 DATA251,133,251,165,203,201,3
3350 DATA208,4,169,24,208,18,201
3360 DATA6,208,4,169,16,208,10
3370 DATA201,5,208,4,169,8,208
3380 DATA2,169,0,24,101,251,133
3390 DATA251,160,0,169,54,133,1
3400 DATA177,251,240,8,153,119,2
3410 DATA200,192,8,208,244,132,198
3420 DATA169,55,133,1,165,203,133
3430 DATA197,173,141,2,141,142,2
3440 DATA96,165,153,208,4,165,157
3450 DATA208,3,76,87,241,165,211
3460 DATA133,202,165,214,133,201,152
3470 DATA72,138,72,165,208,240,6
3480 DATA76,58,230,32,22,231,165
3490 DATA198,133,204,141,146,2,240
3500 DATA247,120,165,207,240,12,165
3510 DATA206,174,135,2,160,0,132
3520 DATA207,32,19,234,32,180,229
3530 DATA201,131,208,16,162,9,120
3540 DATA134,198,189,230,236,157,118
3550 DATA2,202,208,247,240,207,201
3560 DATA13,208,3,76,2,230,201
3570 DATA17,208,193,166,214,224,24
3580 DATA240,3,76,15,132,162,24
3590 DATA160,0,24,32,240,255,230
3600 DATA20,208,2,230,21,32,19
3610 DATA166,160,1,177,95,208,16
3620 DATA169,255,133,20,133,21,169
3630 DATA185,160,132,32,30,171,76
3640 DATA18,132,160,2,177,95,133
3650 DATA20,200,177,95,133,21,169
3660 DATA162,141,0,3,169,132,141
3670 DATA1,3,104,141,183,132,104
3680 DATA141,184,132,160,1,132,15
3690 DATA76,215,166,169,139,141,0
3700 DATA3,169,227,141,1,3,173
3710 DATA184,132,72,173,183,132,72
3720 DATA76,18,132,0,0,13,13
3730 DATA18,42,42,42,42,42,42
3740 DATA42,42,42,42,42,42,32
3750 DATA69,78,68,32,79,70,32
3760 DATA80,82,79,71,82,65,77
3770 DATA32,42,42,42,42,42,42
3780 DATA42,42,42,42,42,42,13
3790 DATA0,32,33,171,32,121,0
3800 DATA240,80,240,94,201,163,240
3810 DATA107,201,166,24,240,102,201
3820 DATA238,208,20,160,1,177,122
3830 DATA201,2,208,12,32,115,0
3840 DATA32,115,0,32,139,133,76
3850 DATA233,132,32,121,0,201,44
3860 DATA240,55,201,59,240,97,32
3870 DATA158,173,36,13,48,195,32
3880 DATA221,189,32,135,180,32,33
3890 DATA171,32,59,171,208,184,169
3900 DATA0,157,0,2,162,255,160
3910 DATA1,165,19,208,16,169,13
3920 DATA32,71,171,36,19,16,5
3930 DATA169,10,32,71,171,73,255
3940 DATA96,56,32,240,255,152,56
3950 DATA233,10,176,252,73,255,105
3960 DATA1,208,25,8,56,32,240
```

Basic Loader

```
3970 DATA255,132,9,32,155,183,201
3980 DATA41,240,3,76,8,175,40
3990 DATA144,6,138,229,9,144,5
4000 DATA170,232,202,208,6,32,115
4010 DATA0,76,238,132,32,59,171
4020 DATA208,242,76,30,171,32,12
4030 DATA134,32,250,174,32,121,0
4040 DATA32,49,134,176,8,32,69
4050 DATA134,142,111,134,176,66,32
4060 DATA46,134,176,8,32,78,134
4070 DATA142,112,134,176,53,32,46
4080 DATA134,176,8,32,72,134,142
4090 DATA113,134,176,40,32,46,134
4100 DATA176,8,32,72,134,142,114
4110 DATA134,176,27,32,46,134,176
4120 DATA8,32,72,134,142,115,134
4130 DATA176,14,32,46,134,144,3
4140 DATA76,8,175,32,75,134,142
4150 DATA116,134,32,247,174,173,116
4160 DATA134,240,5,169,147,32,22
4170 DATA231,173,113,134,141,134,2
4180 DATA173,114,134,141,33,208,173
4190 DATA115,134,141,32,208,172,111
4200 DATA134,174,112,134,24,76,240
4210 DATA255,56,32,240,255,140,111
4220 DATA134,142,112,134,173,33,208
4230 DATA141,114,134,173,32,208,141
4240 DATA115,134,173,134,2,141,113
4250 DATA134,169,0,141,116,134,96
4260 DATA32,115,0,201,44,208,2
4270 DATA56,96,201,41,240,2,24
4280 DATA96,104,104,32,115,0,76
4290 DATA230,133,169,40,44,169,16
4300 DATA44,169,2,44,169,25,141
4310 DATA110,134,32,158,183,236,110
4320 DATA134,176,14,32,121,0,201
4330 DATA41,240,211,201,44,240,213
4340 DATA76,8,175,162,14,76,55
4350 DATA164,0,0,0,0,0,0
4360 DATA0,76,113,168,76,8,175
4370 DATA76,8,175,76,8,175,76
4380 DATA8,175,76,8,175,76,8
4390 DATA175,76,8,175,76,8,175
4400 DATA76,8,175,76,8,175,76
4410 DATA8,175,76,8,175,76,8
4420 DATA175,76,8,175,76,8,175
4430 DATA76,8,175,76,8,175,76
4440 DATA8,175,76,8,175,76,8
4450 DATA175,76,8,175,76,8,175
4460 DATA76,8,175,76,8,175,76
4470 DATA8,175,76,8,175,76,8
4480 DATA175,76,8,175,76,8,175
4490 DATA76,8,175,136,83,30,255,-1
```

Machine Code Listing

```
LINE

 .LIB   INITRT
 *      =$8000
        .WOR COLD        ;COLD START ENTRY
        .WOR WRST        ;RESTORE ENTRY
        .BYT $C3,$C2,$CD,'80'


;
LINK    .WOR $E38B
        .WOR $A483
        .WOR CRNCHT
        .WOR PRINT
        .WOR HANDLE
        .WOR ARITH
;
VECTOR  JMP $B248        ;USR JUMP
        .BYT 0
        .WOR $EA31       ;IRQ
        .WOR WRST01      ;BREAK
        .WOR $FE47       ;NMI
        .WOR $F34A       ;OPEN
        .WOR $F291       ;CLOSE
        .WOR $F20E       ;SET INPUT
        .WOR $F250       ;SET OUTPUT
        .WOR $F333       ;RESTORE I/O
        .WOR LISTER      ;INPUT
        .WOR $F1CA       ;OUTPUT
        .WOR $F6ED       ;TEST-STOP
        .WOR $F13E       ;GET
        .WOR $F32F       ;ABORT I/O
        .WOR WRST01      ;WARM RESTART
        .WOR $F4A5       ;LOAD
        .WOR $F5ED       ;SAVE
;
WRST    JSR $F6BC        ;UPDATE TIME
        JSR $FFE1        ;STOP KEY?
        BEQ WRST01       ;YES
        JMP $FE72        ;NO
WRST01  JSR $FDA3        ;INIT I/O
        JSR $E518        ;INIT VIC CHIP
        JSR SETKER       ;INIT KERNAL VECTORS
        JSR $FFCC        ;RESTORE I/O
        LDA #$00
        STA $13          ;INPUT PROMPT FLAG
        JSR $A67A        ;INIT BASIC
        CLI              ;ENABLE IRQ
WRST02  LDX #$80         ;SET FOR READY
        JMP $E388        ;GO TO READY
;
SETKER  LDX #<VECTOR     ;POINT TO
        LDY #>VECTOR     ;KERNAL VECTORS
        STX $C3
        STY $C4
        LDY #$23         ;LOOP TO COPY VECTORS

STKER1  LDA ($C3),Y      ;GET BYTE
        STA $0310,Y      ;STORE IT
        DEY
        BPL STKER1       ;AND NEXT
        LDA #<FUNC       ;POINT TO FUNCTION
        LDY #>FUNC       ;KEY ROUTINE
        STA $028F        ;STORE IN KEYBOARD
        STY $0290        ;TABLE SETUP VECTOR
        RTS
;
COLD    STX $D016        ;SHRINK SCREEN
        JSR $FDA3        ;INIT I/O
        JSR $FD50        ;INIT SYSTEM CONSTANTS
        JSR $FF5B
        JSR SETKER       ;SET KERNAL VECTORS
        CLI              ;ENABLE IRQ
        JSR SETBAS       ;SET BASIC VECTORS
        JSR $E3BF        ;INIT BASIC
        LDA #$80         ;SET TOP OF RAM
        STA $34
        STA $36
        STA $38
        LDA #$00
        STA $33
        STA $35
```

```
        STA $37
        LDA #<POWER        ;POINT TO POWER
        LDY #>POWER        ;UP MESSAGE
        JSR $E42D          ;OUTPUT MESSAGE
        LDX #$FB
        TXS                ;SET STACK POINTER
        BNE WRST02         ;ALWAYS
;
POWER   .BYT $93,$0D

        .BYT '     **** EXTENDED 64 BASIC'
        .BYT ' V01 ****',$0D,$0D


        .BYT ' 64K RAM SYSTEM  ',$00

;
SETBAS  LDX #$0B           ;LOOP
STBAS1  LDA LINK,X         ;GET BYTE
        STA $0300,X        ;STORE IT
        DEX
        BPL STBAS1         ;DO NEXT
        RTS
;
CLIST   .BYT 'RU',$CE

        .BYT 'CT',$CC

        .BYT 'APPEN',$C4

        .BYT 'AUT',$CF

        .BYT 'CATALO',$C7

        .BYT 'CHANG',$C5

        .BYT 'CHAI',$CE

        .BYT 'CRUNC',$C8

        .BYT 'DELET',$C5

        .BYT 'DIS',$CB

        .BYT 'DOK',$C5

        .BYT 'DUM',$D0

        .BYT 'EXE',$C3

        .BYT 'FIN',$C4

        .BYT 'GE',$D4

        .BYT 'KE',$D9

        .BYT 'MA',$D4

        .BYT 'MERG',$C5

        .BYT 'OL',$C4

        .BYT 'PO',$D0

        .BYT 'PU',$D4

        .BYT 'RENUMBE',$D2

        .BYT 'REPEA',$D4

        .BYT 'SOR',$D4

        .BYT 'TRACEO',$CE

        .BYT 'TRACEOF',$C6

        .BYT 'TYP',$C5

        .BYT 'UNTI',$CC


        .BYT 'DEE',$CB

        .BYT 'HIME',$CD

        .BYT 'LOME',$CD

        .BYT 'VARPT',$D2

        .BYT 0
;
CADDR   .WOR RUN-1
        .WOR CTL-1
        .WOR APPEND-1
        .WOR AUTONO-1
        .WOR CATLOG-1
        .WOR CHANGE-1
        .WOR CHAIN-1
        .WOR CRUNCH-1
        .WOR DELETE-1
        .WOR DISK-1
        .WOR DOKE-1
        .WOR DUMP-1
        .WOR EXEC-1
        .WOR FIND-1
        .WOR GET-1
        .WOR KEY-1
        .WOR MAT-1
        .WOR MERGE-1
        .WOR OLD-1
        .WOR POP-1
        .WOR PUT-1
        .WOR RENUMB-1
        .WOR REPEAT-1
        .WOR SORT-1
        .WOR TRON-1
        .WOR TROFF-1
        .WOR TYPE-1
        .WOR UNTIL-1
;
        .WOR DEEK-1
        .WOR HIMEM-1
        .WOR LOMEM-1
        .WOR VARPTR-1
;
FNSTRT =29
.END
.LIB   CRUNCH-TOKEN
; CRUNCH KEYWORD LINK
;      FOR USE WITH THE ROUTINES IN
; 'ADVANCED COMMODORE 64 BASIC REVEALED'
;
CRNCHT  LDX $7A
        LDY #$04
        STY $0F
CRNC01  LDA $0200,X        ;GET CHAR
        BPL CRNC02         ;CHAR IS OK
        CMP #$FF           ;PIPRINT
        BEQ CRNC08         ;YES,SEND IT
        INX                ;NO, ILLEGAL CHAR
        BNE CRNC01         ; SO DO NEXT
;
CRNC02  CMP #$20           ;SPACEPRINT
        BEQ CRNC08         ;YES, SEND IT
        STA $08
        CMP #$22           ;QUOTESPRINT
        BEQ CRNC12         ;YES, SCAN QUOTE END
        BIT $0F
        BVS CRNC08         ;SEND CHAR
        CMP #$3F           ;'PRINT' PRINT
        BNE CRNC03         ;NO
        LDA #$99           ;SET TO PRINT TOKEN
        BNE CRNC08         ;SEND IT
;
CRNC03  CMP #$30           ;<0 PRINT
        BCC CRNC04         ;YES, HUNT FOR KEYWORD
        CMP #$3C           ;< '<' PRINT
        BCC CRNC08         ;YES, SEND CHAR
CRNC04  JMP CRNC15         ;HUNT FOR KEYWORD
;
CRNC05  LDA #$EE           ;ONE OF MINE
        .BYT $2C           ;SKIP NEXT 2 BYTES
;
CRNC06  ORA $0B            ;ONE OF BASIC'S
CRNC07  LDY $71            ;RESTORE Y
CRNC08  INX                ;NEXT POSITION
```

Machine Code Listing

```
        INY
        STA $01FB,Y     ;STORE IT
        CMP #$EE        ;MINEPRINT
        BEQ CRNC14      ;YES, SEND 2ND BYTE
        LDA $01FB,Y     ;NO, END OF INPUTPRINT
        BEQ CRNC13      ;YES
        SEC
        SBC #$3A        ;':' PRINT
        BEQ CRNC09      ;YES
        CMP #$49        ;DATA ?
        BNE CRNC10      ;NO
CRNC09  STA $0F
CRNC10  SEC
        SBC #$55        ;REM ?
        BNE CRNC01      ;NO DO NEXT CHAR
        STA $08         ;SET QUOTE FLAG
CRNC11  LDA $0200,X     ;GET BYTE
        BEQ CRNC08      ;END OF INPUT, SEND
        CMP $08         ;QUOTE FLAGPRINT
        BEQ CRNC08      ;YES, SEND
CRNC12  INY             ;STORE CHAR
        STA $01FB,Y
        INX
        BNE CRNC11      ;DO NEXT
;
CRNC13  STA $01FD,Y     ;STORE ZERO
        DEC $7B
        LDA #$FF
        STA $7A
        RTS             ;EXIT CRUNCH
;
CRNC14  LDA $0B         ;GET 2ND BYTE
        INY
        STA $01FB,Y     ;STORE IT
        JMP CRNC01      ;DO NEXT BYTE
;
CRNC15  STY $71         ;SAVE OFF Y
        LDY #$FF
        STX $7A         ; AND X POINTERS
        DEX
        LDA #$01        ;START TOKEN VAL=1
        STA $0B
CRNC16  INY
        INX
CRNC17  LDA $0200,X     ;GET BYTE
        SEC
        SBC CLIST,Y     ;AS KEYWORD TABLEPRINT
        BEQ CRNC16      ;YES, CHECK NEXT
        CMP #$80        ;SHIFT OUTPRINT
        BEQ CRNC05      ;YES, FOUND
        LDX $7A         ;RESTORE BUFFER POINTER
        INC $0B         ;NEXT TOKEN
CRNC18  INY
        LDA CLIST-1,Y   ;END OF KEYWORDPRINT
        BPL CRNC18      ;NO
        LDA CLIST,Y     ;END OF TABLEPRINT
        BNE CRNC17      ;NO, CHECK NEXT
        LDY #$00        ;START TOKEN AT 0
        STY $0B         ;FOR BASIC
        DEY
        LDX $7A         ;GET INPUT POINTER
        DEX
CRNC19  INY
        INX
CRNC20  LDA $0200,X     ;GET BYTE
        SEC
        SBC $A09E,Y     ;AS IN TABLEPRINT
        BEQ CRNC19      ;YES, CHECK NEXT
        CMP #$80        ;SHIFT OUTPRINT
        BNE CRNC21      ;NO, TRY NEXT WORD
        JMP CRNC06      ;YES, SEND BASIC TOKEN
CRNC21  LDX $7A         ;RESTORE INPUT POINTER
        INC $0B         ;NEXT TOKEN
CRNC22  INY
        LDA $A09D,Y     ;END OF WORDPRINT
        BPL CRNC22      ;NO
        LDA $A09E,Y     ;END OF TABLEPRINT
        BNE CRNC20      ;NO, TRY NEXT WORD
        LDA $0200,X     ;ELSE SEND BYTE
        JMP CRNC07
.END
.LIB    PRINT-TOKEN
; PRINT TOKENS LINK
;     FOR USE WITH THE ROUTINES IN
; 'ADVANCED COMMODORE 64 BASIC REVEALED'
;
PRINT   BMI PRIN02      ;A TOKEN
PRIN01  JMP $A6F3       ;PRINT IT
PRIN02  CMP #$FF        ;IS IT PI?
        BEQ PRIN01      ;YES
        BIT $0F         ;QUOTES?
        BMI PRIN01      ;YES
        CMP #$EE        ;ONE OF MINE?
        BEQ PRIN08      ;DO MINE
        JSR PRIN09      ;DO BASIC
        BMI PRIN13      ;ALWAYS
PRIN08  JSR PRIN03      ;DO MINE
PRIN13  JMP $A6EF       ;AND NEXT
;
PRIN03  INY             ;GET TOKEN CHAR
        LDA ($5F),Y
        TAX
        STY $49         ;SAVE Y
        LDY #$FF
PRIN04  DEX
        BEQ PRIN06      ;FOUND IT
PRIN05  INY
        LDA CLIST,Y     ;GET CHAR FROM TABLE
        BPL PRIN05      ;UNTIL END OF WORD
        BMI PRIN04      ;FOUND END OF WORD
PRIN06  INY
        LDA CLIST,Y     ;GET CHAR FROM TABLE
        BMI PRIN07      ;LAST CHAR OF WORD
        JSR $FFD2       ;PRINT IT
        BNE PRIN06      ;NEXT CHAR
PRIN07  RTS             ;DO LAST
;
PRIN09  SEC
        SBC #$7F        ;REMOVE SHIFT
        TAX
        STY $49         ;SAVE .Y
        LDY #$FF
PRIN10  DEX
        BEQ PRIN12      ;FOUND IT
PRIN11  INY
        LDA $A09E,Y     ;GET CHAR FROM TABLE
        BPL PRIN11      ;UNTIL END OF WORD
        BMI PRIN10      ;FOUND END OF WORD
PRIN12  INY
        LDA $A09E,Y     ;GET CHAR FROM TABLE
        BMI PRIN07      ;LAST CHAR OF WORD
        JSR $FFD2       ;PRINT CHAR
        BNE PRIN12      ;ALWAYS
.END
.LIB    HANDLE-TOKEN
; EXECUTE STATEMENT LINK
;     FOR USE WITH THE ROUTINES IN
; 'ADVANCED COMMODORE 64 BASIC REVEALED'
;
HANDLE  JSR $0073       ;GET CODE
        CMP #$EE        ;IS IT MY TOKEN?
        BEQ HAND01      ;YES, DO IT
        CMP #$99        ;IS IT PRINT?
        BEQ DOPRNT      ;YES
        JSR $0079       ;GET CURRENT CHAR
        JMP $A7E7       ;DO BASIC CODE
;
HAND01  JSR HAND02      ;EXECUTE THE CODE
        JMP $A7AE       ;AND NEXT
HAND02  INC $7A         ;GET TOKEN CHAR
        BNE HAND04
        INC $7B
HAND04  LDY #$00
        LDA ($7A),Y     ;GET INPUT BYTE
;
        SEC
        SBC #$01
        ASL A           ;TIMES 2
        TAY
        LDA CADDR+1,Y   ;GET HI BYTE
        PHA             ;TO STACK
        LDA CADDR,Y     ;GET LO BYTE
        PHA             ;TO STACK
        JMP $0073       ;EXECUTE IT
```

Machine Code Listing

```
;PRINT SPECIAL CASE
;
DOPRNT JSR HAND03          ;DO PRINT COMMAND
       JMP $A7AE           ;DO NEXT COMMAND
;
HAND03 LDA PADDR+1         ;GET HI BYTE
       PHA                 ;TO STACK
       LDA PADDR           ;GET LO BYTE
       PHA                 ;TO STACK
       JMP $0073           ;EXECUTE PRINT
PADDR  .WOR PRINTT-1       ;VECTOR FOR PRINT
.END
.LIB   ARITH-TOKEN
; ARITHMETIC LINK
;      FOR USE WITH THE ROUTINES IN
; 'ADVANCED COMMODORE 64 BASIC REVEALED'
;
ARITH  LDA #$00            ;TYPE FLAG TO NUMERIC
       STA $0D
       JSR $0073           ;GET BYTE
       CMP #$EE            ;ONE OF MINE?
       BEQ ARITH1          ;YES
       JSR $0079           ;GET CURRENT CHAR
       JMP $AE8D           ;OPERATE
;
ARITH1 INC $7A             ;GET TOKEN CHAR
       BNE ARITH3
       INC $7B
ARITH3 LDY #$00
       LDA ($7A),Y         ;GET INPUT BYTE
       CMP #FNSTRT         ;IS IT A FUNCTION
       BCS ARITH2          ;YES
       JMP $AF08           ;SYNTAX ERROR
;
ARITH2 STA $24             ;SAVE TOKEN VAL
       LDA #$AD            ;SETUP RETURN ADDRESS
       PHA
       LDA #$8C
       PHA
       DEC $24
       LDA $24             ;GET TOKEN
       ASL A               ;TIMES 2
       TAX
       LDA CADDR+1,X       ;GET HI BYTE
       PHA
       LDA CADDR,X         ;GET LO BYTE
       PHA
       JMP $0073           ;EXECUTE FUNCTION
.END
.LIB   FUNC-KEYS
FUNC   LDA $9D             ;DIRECT?
       BEQ FUNC01          ;NO
       LDA #$01            ;QUOTES?
       BIT $D4
       BNE FUNC01          ;YES, IGNORE
       LDA $CB             ;KEY PRESSED
       CMP #$03            ;F7?
       BCC FUNC01          ;NO, LESS THAN
       CMP #$07            ;F5?
       BCC FUNC02          ;YES, IS A FUNCTION KEY
FUNC01 JMP $EB48           ;DO NORMAL KEYS
;
FUNC02 CMP $C5             ;ALREADY DONE?
       BEQ FUNC01          ;YES
       LDA #$00            ;CLEAR POINTER
       STA $FC
       STA $FB
       LDA #$01            ;SHIFT KEY?
       BIT $028D
       BEQ FUNC03          ;NO
       LDA #$20
       STA $FB
FUNC03 LDA #$BF            ;ADD START OF STORE
       STA $FC             ; TO POINTER
       LDA #$C0
       CLC
       ADC $FB
       STA $FB
       LDA $CB
       CMP #$03            ;F7?
       BNE FUNC04          ;NO
       LDA #24
       BNE FUNC07
FUNC04 CMP #$06            ;F5?
       BNE FUNC05          ;NO
       LDA #16
       BNE FUNC07
FUNC05 CMP #$05            ;F3?
       BNE FUNC06          ;NO
       LDA #8
       BNE FUNC07
FUNC06 LDA #$00            ;MUST BE F1
FUNC07 CLC                 ;SET VAL INTO POINTER
       ADC $FB
       STA $FB
       LDY #$00
       LDA #$36            ;SWITCH OUT BAS ROM
       STA $01
FUNC08 LDA ($FB),Y         ;GET CHAR
       BEQ FUNC09          ;ZERO BYTE TERMINATOR
       STA $0277,Y         ;STORE IN BUFFER
       INY
       CPY #$08            ;ALL 8?
       BNE FUNC08          ;NOT YET
FUNC09 STY $C6             ;#CHARS IN BUFFER
       LDA #$37            ;PUT BASIC ROM BACK
       STA $01
       LDA $CB             ;SET LAST=PRESENT
       STA $C5             ; KEYS,
       LDA $028D           ; SHIFT COMBO
       STA $028E
       RTS                 ;ALL DONE
.END
.LIB   LISTER
LISTER LDA $99
       BNE LIST01          ;NOT KEYBOARD
       LDA $9D
       BNE LIST02          ;IS DIRECT INPUT
LIST01 JMP $F157           ;DO NORMAL
;
LIST02 LDA $D3             ;SAVE CURRENT CURSOR
       STA $CA             ; COLUMN
       LDA $D6
       STA $C9             ; AND ROW
       TYA                 ;SAVE .X AND .Y
       PHA
       TXA
       PHA
       LDA $D0             ;SCREEN OR KEYBOARD?
       BEQ LIST04          ;KEYBOARD
       JMP $E63A           ;DO FOR SCREEN
;
LIST03 JSR $E716           ;DISPLAY CHAR TO SCREEN
LIST04 LDA $C6             ;ANY CHARS IN BUFFER?
       STA $CC             ;IF NOT, BLINK CURSOR
       STA $0292           ;AUTO SCROLL DOWN
       BEQ LIST04          ;REPEAT UNTIL CHAR
       SEI                 ;DISABLE KEYBOARD
       LDA $CF             ;CURSOR BLINK?
       BEQ LIST05          ;NO
       LDA $CE             ;RESTORE ORIGINAL CHAR
       LDX $0287           ; AND COLOUR
       LDY #$00
       STY $CF             ;SWITCH OFF BLINK
       JSR $EA13           ;RESTORE
LIST05 JSR $E5B4           ;REMOVE CHAR FROM BUFFER
       CMP #$83            ;RUN/STOP?
       BNE LIST07          ;NO
       LDX #$09            ;COPY TEXT INTO BUFFER
       SEI
       STX $C6
LIST06 LDA $ECE6,X
       STA $0276,X
       DEX
       BNE LIST06          ;REPEAT UNTIL ALL DONE
       BEQ LIST04          ;DONE, OPERATE ON RUN/STOP
LIST07 CMP #$0D            ;CARRIAGE RETURN?
       BNE LIST08          ;NO
       JMP $E602           ;END OF INPUT
;
LIST08 CMP #$11            ;CURSOR DOWN?
       BNE LIST03          ;NO GET NEXT CHAR
       LDX $D6
       CPX #24             ;SCROLL SCREEN?
```

Machine Code Listing

```
       BEQ LIST09       ;YES
       JMP LIST03       ;NO, NEXT CHAR
LIST09 LDX #24          ;SET CURSOR TO
       LDY #$00         ; BEGINNING OF LINE
       CLC
       JSR $FFF0
       INC $14          ;FIND NEXT LINE TO
       BNE LIST10       ; LIST
       INC $14+1
LIST10 JSR $A613        ;GET ADDRESS
       LDY #$01
       LDA ($5F),Y      ;END OF PROGRAM?
       BNE LIST11       ;NO
       LDA #$FF
       STA $14          ;NEXT LINE NUMBER=0
       STA $14+1
       LDA #<EOPMES     ;TELL USER THAT THE
       LDY #>EOPMES     ; END OF PROGRAM HAS
       JSR $AB1E        ; BEEN REACHED
       JMP LIST04       ;GET NEXT CHAR
LIST11 LDY #$02         ;GET LINE NUMBER
       LDA ($5F),Y      ;LO BYTE
       STA $14
       INY
       LDA ($5F),Y      ;HI BYTE
       STA $14+1
       LDA #<LIST12     ;RETURN TO LIST12
       STA $0300        ;AFTER LIST
       LDA #>LIST12
       STA $0301
       PLA              ;SAVE 2 BYTES IN
       STA STACK        ; SAFE LOCATION
       PLA
       STA STACK+1
       LDY #$01
       STY $0F
       JMP $A6D7        ;LIST LINE
LIST12 LDA #$8B         ;RESET ERROR VECTOR
       STA $0300
       LDA #$E3
       STA $0301
       LDA STACK+1      ;RESTORE 2 BYTES
       PHA
       LDA STACK
       PHA
       JMP LIST04       ;DO NEXT CHAR
STACK  .WOR 0
EOPMES .BYT $0D,$0D,$12

       .BYT '************ END OF PROGRAM ************'
       .BYT $0D,$00
 .LIB   PRINT
 PRNT01 JSR $AB21       ;PRINT STRING
 PRNT02 JSR $0079       ;GET CURRENT CHAR
 PRINTT BEQ PRNT05      ;CARRIAGE RETURN
 PRNT03 BEQ PRNT07      ;SEMICOLON
        CMP #$A3        ;TAB?
        BEQ TAB         ;YES
        CMP #$A6        ;SPC?
        CLC
        BEQ TAB         ;YES
        CMP #$EE        ;MINE?
        BNE PRNT08      ;NO
        LDY #$01
        LDA ($7A),Y     ;GET TOKEN
        CMP #$02        ;CTL?
        BNE PRNT08      ;NO
        JSR $0073
        JSR $0073       ;GET NEXT CHAR
        JSR CTL         ;DO CTL
        JMP PRNT02
 ;
 PRNT08 JSR $0079       ;GET CURRENT CHAR
        CMP #$2C        ;','?
        BEQ PRNT09      ;YES
        CMP #$3B        ;';'?
        BEQ TAB04       ;YES
        JSR $AD9E       ;EVALUATE EXPRESSION
        BIT $0D         ;WHICH TYPE?
        BMI PRNT01      ;STRING
        JSR $BDDD       ;CONVERT FAC#1 TO STRING
        JSR $B487
        JSR $AB21
        JSR $AB3B
        BNE PRNT02
PRNT04  LDA #$00
        STA $0200,X
        LDX #$FF
        LDY #$01
        LDA $13
        BNE PRNT07
PRNT05  LDA #$0D        ;CARRIAGE RETURN
        JSR $AB47
        BIT $13
        BPL PRNT06      ;FILE#>128 NO LF
        LDA #$0A        ;LINE FEED
        JSR $AB47       ;PRINT IT
PRNT06  EOR #$FF
PRNT07  RTS
;
;DECIMAL TABLUATOR
;
PRNT09  SEC
        JSR $FFF0       ;GET CURSOR POS
        TYA
        SEC
PRNT10  SBC #$0A        ;MINUS 10
        BCS PRNT10
        EOR #$FF
        ADC #$01
        BNE TAB01
;
;TAB AND SPC
;
TAB     PHP
        SEC
        JSR $FFF0       ;GET CURSOR POSITION
        STY $09         ;STORE IN TEMP
        JSR $B79B       ;GET 1 BYTE PAR
        CMP #$29        ;')'?
        BEQ TAB10       ;YES
        JMP $AF08       ;SYNTAX ERROR
TAB10   PLP             ;TAB OR SPC?
        BCC TAB02       ;SPC
        TXA             ;TAB VALUE
        SBC $09         ;MINUS COLUMN POSITION
        BCC TAB04       ;LESS THAN
TAB01   TAX
TAB02   INX
TAB03   DEX
        BNE TAB05
TAB04   JSR $0073       ;GET NEXT CHAR
        JMP PRNT03      ;BACK TO PRINT
TAB05   JSR $AB3B       ;OUTPUT SPACE/RIGHT
        BNE TAB03       ;ALWAYS
        JMP $AB1E
.END

.LIB    CTL
CTL     JSR CTLDEF      ;SET DEFAULT
        JSR $AEFA       ;SCAN '('
        JSR $0079       ;GET CURRENT CHAR
        JSR CHECKN+3    ;NEXT PAR?
        BCS CTL01       ;NO
        JSR GV1         ;GET VALUE
        STX CTXPOS      ;STORE IT
        BCS CTLEN1      ;FOLLOWED BY ')'
CTL01   JSR CHECKN      ;NEXT PAR?
        BCS CTL02       ;NO
        JSR GV4         ;GET VALUE
        STX CTYPOS      ;STORE IT
        BCS CTLEN1      ;FOLLOWED BY ')'
CTL02   JSR CHECKN      ;NEXT PAR?
        BCS CTL03       ;NO
        JSR GV2         ;GET VALUE
        STX CTCUR       ;STORE IT
        BCS CTLEN1      ;FOLLOWED BY ')'
CTL03   JSR CHECKN      ;NEXT PAR?
        BCS CTL04       ;NO
        JSR GV2         ;GET VALUE
        STX CTSC        ;STORE IT
        BCS CTLEN1      ;FOLLOWED BY ')'
CTL04   JSR CHECKN      ;NEXT PAR?
```

Machine Code Listing

```
        BCS CTL05       ;NO
        JSR GV2         ;GET VALUE
        STX CTBD        ;STORE IT
        BCS CTLEN1      ;FOLLOWED BY ')'
CTL05   JSR CHECKN      ;NEXT PAR?
        BCC CTL06       ;YES
        JMP $AF08       ;COMMA, SYNTAX ERROR
CTL06   JSR GV3         ;GET VALUE
        STX CTCFLG      ;STORE IT
CTLEN1  JSR $AEF7       ;SCAN ')'
;
CTLEND  LDA CTCFLG      ;CLEAR SCREEN?
        BEQ CTEND1      ;NO
        LDA #147        ;CHAR FOR CLS
        JSR $E716       ;OUTPUT TO SCREEN
CTEND1  LDA CTCUR       ;GET CURSOR COLOUR
        STA $0286       ;SET IT
        LDA CTSC        ;GET SCREEN COLOUR
        STA $D021       ;SET IT
        LDA CTBD        ;GET BORDER COLOUR
        STA $D020       ;SET IT
        LDY CTXPOS      ;GET X POSITION
        LDX CTYPOS      ;GET Y POSITION
        CLC             ;FLAG WRITE
        JMP $FFF0       ;SET CURSOR POS AND EXIT
;
CTLDEF  SEC             ;FLAG READ
        JSR $FFF0       ;GET CURSOR POS
        STY CTXPOS      ;STORE X
        STX CTYPOS      ;STORE Y
        LDA $D021       ;GET SCREEN COLOUR
        STA CTSC        ;STORE IT
        LDA $D020       ;GET BORDER COLOUR
        STA CTBD        ;STORE IT
        LDA $0286       ;GET CURSOR COLOUR
        STA CTCUR       ;STORE IT
        LDA #$00        ;ZERO SCREEN CLEAR
        STA CTCFLG      ;FLAG
        RTS
;
CHECKN  JSR $0073       ;GET NEXT CHAR
        CMP #$2C        ;IS IT A COMMA?
        BNE CHECKB      ;NO
CHECKS  SEC             ;FLAG FOR COMMA
        RTS
CHECKB  CMP #$29        ;IS IT ')'?
        BEQ CHECKA      ;YES, DONE
CHECKC  CLC             ;SET NO COMMA
        RTS
CHECKA  PLA             ;REMOVE RTS
        PLA             ;ADDRESS
        JSR $0073       ;GET NEXT CHAR
        JMP CTLEND      ;SET VALUES
;
GV1     LDA #40         ;COMPARE X POS
        .BYT $2C        ;SKIP
GV2     LDA #16         ;COMPARE COLOUR
        .BYT $2C        ;SKIP
GV3     LDA #2          ;COMPARE CLEAR FLAG
        .BYT $2C        ;SKIP
GV4     LDA #25         ;COMPARE Y POS
        STA VCOMP       ;STORE COMPARE VALUE
        JSR $B79E       ;GET 1 BYTE#
        CPX VCOMP       ;IN RANGE 0-(VCOMP-1)
        BCS GERR        ;NO
        JSR $0079       ;GET CURRENT CHAR
        CMP #$29        ;IS IT ')'
        BEQ CHECKS      ;YES, FLAG END
        CMP #$2C        ;IS IT ','
        BEQ CHECKC      ;YES FLAG ANOTHER
        JMP $AF08       ;SYNTAX ERROR
GERR    LDX #$0E        ;ILLEGAL QUANTITY
        JMP $A437       ;SEND ERROR
;
VCOMP   .BYT 0          ;VALUE COMPARE
CTXPOS  .BYT 0          ;X POSITION
CTYPOS  .BYT 0          ;Y POSITION
CTCUR   .BYT 0          ;CURSOR COLOUR
CTSC    .BYT 0          ;SCREEN COLOUR
CTBD    .BYT 0          ;BORDER COLOUR
CTCFLG  .BYT 0          ;CLEAR SCREEN FLAG
.END
.LIB    DUMMY
RUN     JMP $A871
APPEND  JMP $AF08
AUTONO  JMP $AF08
CATLOG  JMP $AF08
CHANGE  JMP $AF08
CHAIN   JMP $AF08
CRUNCH  JMP $AF08
DELETE  JMP $AF08
DISK    JMP $AF08
DOKE    JMP $AF08
DUMP    JMP $AF08
EXEC    JMP $AF08
FIND    JMP $AF08
GET     JMP $AF08
KEY     JMP $AF08
MAT     JMP $AF08
MERGE   JMP $AF08
OLD     JMP $AF08
POP     JMP $AF08
PUT     JMP $AF08
RENUMB  JMP $AF08
REPEAT  JMP $AF08
SORT    JMP $AF08
TRON    JMP $AF08
TROFF   JMP $AF08
TYPE    JMP $AF08
UNTIL   JMP $AF08
DEEK    JMP $AF08
HIMEM   JMP $AF08
LOMEM   JMP $AF08
VARPTR  JMP $AF08
.END
.END
```



Dave Crisp has been looking at serious business software from Gemini — here's his verdict.

THIS LITTLE OFFERING COMES FROM Gemini a company purporting to be the home of "serious software". The title suggests that Gemini has more business goodies in the pipeline and the "serious software" slogan bears some thinking about. Looking up "serious" in my dictionary gives me synonyms such as "grave" – perhaps a portent of the consequences of running the software!

Anyway brass tacks first...

Business Pack No 1 contains a Cashbook system, a VAT file facility and a Final Accounts package. A pretty comprehensive trio of programs for anyone who needs to keep accounts.

The pack comes complete with three manuals/user guides. These are quite well presented if a little light on content and obviously devised by someone who knows accountancy and computers a great deal better than he does the users.

My accountant was quite impressed with the facilities offered by the package, but threw up his hands in horror when I explained that it was designed for use by small businesses with little or no expertise in either book-keeping or data processing.

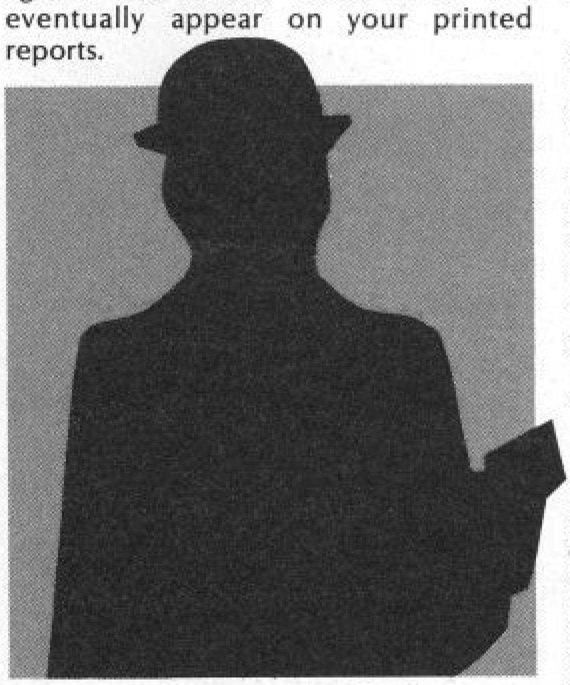
The review copy came on disk and loaded painlessly enough, albeit slowly. The tape-based version comes on three separate cassettes and, presumably takes long enough to LOAD to allow you to pop down to the Dog & Diskdrive for pint.

Finding a spelling mistake in the opening paragraph of the operating instruction notes does nothing to improve sagging confidence, but plough on regardless...

Each session begins by requesting you to enter a date. Remember its accuracy is vital to the end result. The system calculates the day of the week for you – but take care!

Various prompts appear to assist you as the session progresses. Obviously when setting the system up from scratch there are functions to carry out which won't be required again until the start of a new accounting year. The first of these (and potentially the most dangerous) is prompted by "Initialise a new file?". A "Y" response requests confirmation. At this stage if you respond in the affirmative all account balances are cleared down – not much fun if you are halfway through your financial year!

Cashbook entries are restricted to 25 transactions per batch, whilst analyses and posting totals are only allowed to five separate accounts. If you are inputting a lot of data it is wise to do some pre-processing analysis of your own (i.e. sort different types of transactions into separate piles). Gemini recommends labelling each batch with a header containing details of the entries – imperative if you are to have any check against the various totals which will eventually appear on your printed reports.



Each of the three systems gives you a menu. The options obviously vary according to the particular service you wish to access. Every conceivable alternative is included, but be sure you fully understand the implications of the actions you take – particularly at the Journal entry stage. Single entry journals are frowned on by many accountants. To make sure your's is happy be sure to keep plenty of documentary back-up and remember that there is no automatic entry to Control or VAT memo accounts. Always err on the side of caution.

There are enough report options available to satisfy the most critical of accountants and the output from the VAT File system should be sufficient to allow you to complete your returns without any problem.

The ususal dangers stalk the night so far as printers are concerned – a general problem. Don't switch off in the wrong sequence or while a program is running. Also, if you use anything except a Commodore printer (who doesn't?) you will need to change the ASCII code for £. The relevant code should appear in your printer handbook.

I would strongly recommend that you purchase one of the "Accounting Made Easy" type books and have a good read before using this package.

Until you have some grasp of the principles of double entry book-keeping you will have untold problems. Gemini advises that you approach your accountant prior to preparing profit & loss account and balance sheet – sounds like a good idea.

Two particular aspects of the package are especially poor.

Firstly, paraphrasing the section of the manual referring to data saving (I didn't have the nerve to test it) "if an error occurs while you are saving data, type GOTO 340 and try again" – how polished and professional can you get! When I was a green young programmer if I'd suggested to my project leader tht such a statement should appear in the operator's instructions, I would have been flayed alive with a wire brush.

Secondly if you accidentally hit RUN/STOP and RESTORE simultaneously you automatically lose any data input during the session.

I find it difficult to recommend Combination Business Pack No 1 as there are a number of better and cheaper alternatives. Gemini still has three important lessons to learn: Do more market research; when the "basic" ideas work – convert them to a faster medium such as machine code; employ good systems analysts and a technical author.

Roll on No 2.

The best part of this particular issue is the lovely lady who appears, smiling, on the front cover. If only someone would plug her C64 into the TV monitor for her.

COMMUNICATION

One of the biggest problems with Compunet is actually finding your way round the system. To help you, here is a list of major areas that are available. If you don't already own a Commodore modem just look at what you're missing.



* = a 'JUNGLE' where users can upload

Adventure games (telesoftware)	2313
*Advice Centre	1204
*Amateur Radio CUG	112258
Arcade games (telesoftware)	2311
Arcade adventures (telesoftware)	2312
Armchair sport (telesoftware)	2316
*Art Gallery	1700
Board & Logic (telesoftware)	2314
*Bogg's Music Requests	120359
Business (Briefcase)	800
*Business Jungle	8000
Business telesoftware	820
Business software index	820
Business services	840
*Business software (user)	1260
*Careers helpline (MSC)	2102
Careers advice (MSC)	540
*Cartoon Corner	131229
*Chat-On-Line	1801
*Chess (mailbox)	113880
*Club 64	121674
*Clubs & Societies	1400
*CND64	122240
Commodore	210
How to Contact	112236
News	112227
Service	112749
Area Dealer Index	112857
Dealers by Area	112749
Index of Software on Cnet	107495
Technical Tips	112228 & 112229
SoftPost	112222
SoftPost Main Catàlogue	113961
Order Form	106780
Members' Offers	113962
Compunet	120
Compunet communications s/w	109791
Editing	122
Electronic mail guide	140
Guide to Editing	117424
Fees	123
*Helplines	2101
Highlights Index	1 & 150
Hints	107711
How to download	1210
How to upload	1210
News	107707
Uploading	12185
User Guide	120
Utilities (software)	109561
Comp-U-Store (teleshopping)	310
Aftersales	115231
Free trial	108714
Link to Comp-U-Store	111063
Membership agreement	110993

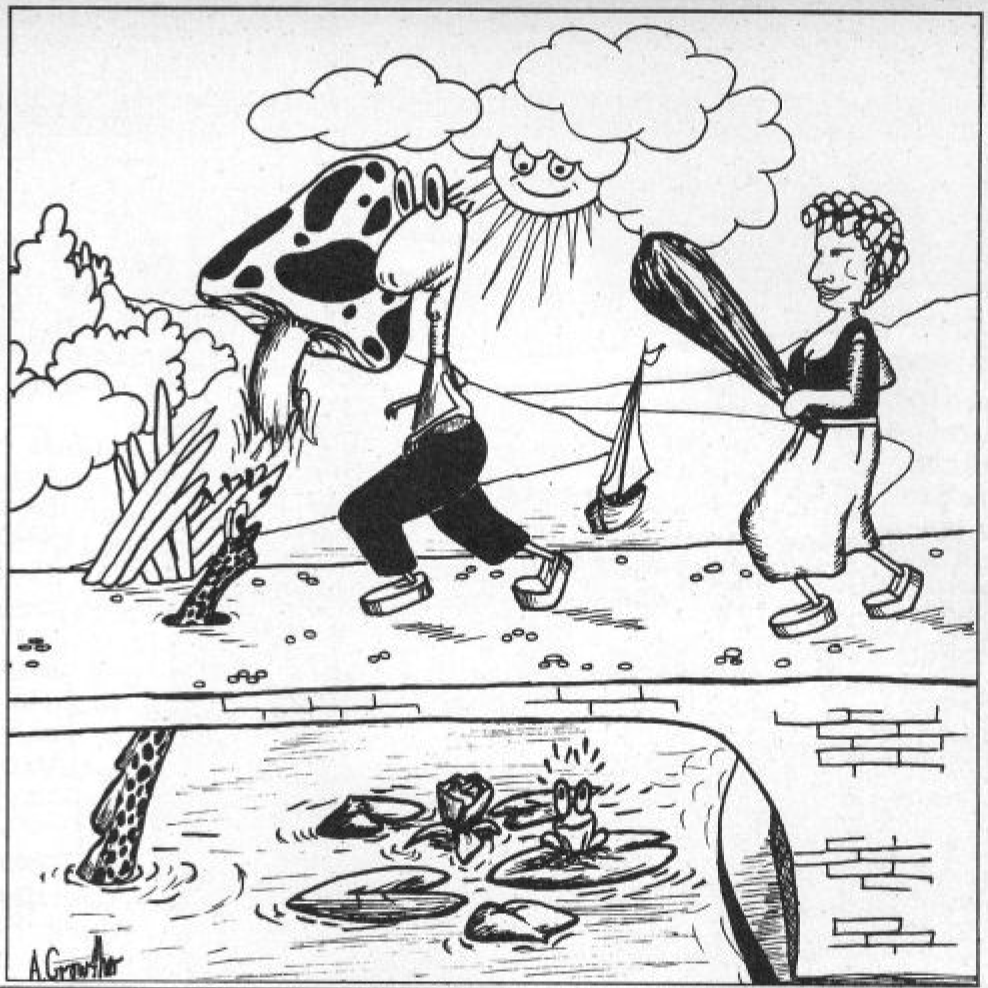
Product news	111682
User Guide	110994
Communications software	2020
*Communications Jungle	2000
Compucat (micro news)	270
Computing software index	240
Computing software (commercial)	240
Computing software (user)	1240
*Dave Martin's Board	110832
Dealer facilities	115390
*Dr Fogg's Spot	111757
Educational Software	530
*Educational Software (user)	1270
Financial News	202
*Games Software (user)	1250
*Games Noticeboard	118943
*Games Spot	1900
*Gof Spot (Virgin Games)	117089
*Hall of Fame	1750
Highlights Index	1 & 150
*Hobbies & Interests	1500
Horoscopes	202
ICC (Independent Compunet Club)	116354
Club details	116535
Committee area	116537
*ICC Jungle	116540
Notice Board	116536
Regional Areas	116538
Software Section	116539
*Street Corner	120457
ICPUG	108692
Committee area	108697
General area	108693
General News	121135
Guide to ICPUG	115459
Legal information	115014
*Members' Board	108698
Members' information	108699
Regional News	114990
Software	113281
Technical tips	108696
*Joint Users Club	118101
*Jungle	600 & 610
Jungle News	117420
Mailbox Gaming	118244
Manpower Services Commission	540
MUD (Multi-User Dungeon)	107923
Link to Mud	108109
*Mud Spot	111628
*Music & graphics	1280
Music & graphics telesoftware	2318
News Headlines	202
*Person to Person	1100
Prestel/Viewdata software	170
*Reviews	1230
Spacerat (micro news)	260
*Schools Jungle	5000
Simulations (telesoftware)	2315
Software Park	230
Softshop	231
Software indexes	150
Software Purchase guarantee	123639
Software suppliers	
Alligata	107512
Bubble Bus	125890
Club64	121674
Commodore	106616
Compunet	240
Creative Sparks (Thorn Emi)	108751
Firebird	132357
Lancaster (Chris)	109766
Llamasoft	108741
Mr Chip	107351
Mr Micro/CBM	106827
Orpheus	133072
PSS	132354
Software suppliers cont.	
Quick-Count	130505
RSAC	121024
Solar Software	108886
Talent	106668
Terminal	125185
Virgin	121675
Your Commodore	290
Your 64	280
Y2	109561
Strategy & War (telesoftware)	2317
Talkback	107703
*Tips and Hints	1600
*Tony's (Crowther) Directory	118951
*Trading Post (small ads)	1300
Update (Cnet news)	107707
*User Noticeboard	1800
User-to-User software	175
Viewtel	202
What's New	107701
*Yakstuff (Llama)	123570
Your Commodore	290
Your 64	280



CORNER

COMPETITION

COMPETITION





Fancy the chance of winning a prize worth £1000? Enter our competition and you could win one of 30 copies of our Game of the Month.

The rules

ENTRIES WILL NOT BE ACCEPTED FROM employees of Argus Specialist Publications, Wizard Development Company Ltd and Alabaster Passmore and Sons. This restriction also applies to employee's families and agents of the companies.

The how to enter section forms part of the rules. The Editor's decision is final and no correspondence will be entered into.

How to enter

There are a few differences between the two pictures. Study them carefully and circle the differences on the picture attached to the coupon. Fill in the coupon clearly and seal the picture and coupon in an envelope. Write clearly the number of differences you found on the back of the envelope.

Post your entry to:
William Competition, Your Commodore, No 1 Golden Square, London W1R 3AB. The closing date is first post on Thursday, 31st October 1985. You may enter as many times as you like but each entry must be on an official coupon – not a copy – and sealed in a separate envelope.

Important: Please follow closely the guidelines on entering – incomplete coupons and entries with no numbers on the back cannot be considered. If you are a winner, the coupon will act as a label for your prize so clear writing is essential.

Name ....
Address ....
....
....
.... Post code........

Number of differences found ....

Send to: William Competition, Your Commodore, No 1 Golden Square, London W1R 3AB. Write clearly and fully and don't forget to put the number of differences you found on the back of the envelope.

Fed up with the speed of Basic? Do you find machine code too complicated? David Janda takes a look at Forth. It may solve your problems.

GOING FORTH



FORTH IS A REMARKABLE LANGUAGE. Not only is it the fastest language available for home micros, but it is also very easy to implement. Just about everything is unique to this language, and many people have used it for various applications.

To get an idea of Forth it is worth looking at its history. Forth was developed in the late 60s by one man called Charles H Moore. Moore was working at the National Radio Astronomy in Kitt Peak Arizona, and was very frustrated with the limitations of existing programming languages.

Over the years, Moore developed a set of programming tools which he could add to when he pleased. He decided to develop his tools into a control language, and Forth was borne. Since then, there have been numerous versions of the language, but basically the choice is between two Forth-77 and Fig-Forth.

The first point to note about Forth is that it doesn't use instructions or commands like Basic. Neither is it restricted to the number of facilities it possesses when purchased. Instead, Forth uses words. Every version comes with a set of words that make up the nucleus of the language.

The set of Forth words is stored in what is called the dictionary. Just like Basic (but more flexible) a Forth program can consists of any number or combination of words. VLIST is a Forth word. Its purpose is to display the contents of the dictionary. About the only other similarity between Basic and Forth is that the latter has an immediate mode where single commands (sorry, words) can be executed. For larger programs, an editor of some sort is used to enter the source code.

The real reason why Forth stands head and shoulders above the rest is its facility to allow the user to add to it. The analogy of someone opening a door will demonstrate this point as well as show the structure of a Forth program:

```
: GRASP GRASP HANDLE ;
: TURN TURN HANDLE ;
: OPEN PULL DOOR TOWARDS YOU ;
: ENTER WALK THROUGH ;

: DOOR GRASP TURN OPEN ENTER ;
```

In this example, there are four words, grasp, turn, open and enter. Each new word consists of a number of Forth words which are terminated by the semi-colon. Finally, a word called DOOR has been defined which consists of the other four words. So, to perform the four words I would simply enter DOOR.

The Stack

Forth has been described by many as the high-level low-level language. The reason for this is because Forth offers such high-level control structures, yet requires the user to maintain the stack. As a matter of fact, the stack (and understanding how to manipulate it) is probably the single most important aspect of learning the language.

The reason for this is because practically every Forth word alters the contents of the stack in some way. So it is entirely up to the user to maintain the stack, as there is no error checking, bound checking or such like. Even though it is a difficult concept to handle at first, stack manipulation does have its rewards. In Forth's case it produces programs that can run at 80% of the speed of machine code programs!

The stack itself is an area of memory that is used as a temporary storage space for numbers, data, and such like. It operates on a first-in last-out basis, and numbers can be added to it (pushed) or removed (popped).

To put the number six on the stack in Forth in immediate mode, the following is done:

```
6 RETURN
```

This places six on the stack. Adding more words is simple:

```
5 2 9 34 21   RETURN
```

Removing the numbers from the stack can be done in a number of ways. By far the simplest is to print them, and this is done with the dot command'.', which removes one item from the stack and displays it:

```
...... RETURN
```

would print:

```
21
34
9
2
5
6
```

The reason for the 'backwardsness' is because the first number placed on the stack will be at the bottom, and hence the last out.

The two methods of placing numbers on the stack are in the immediate mode (as above), or within a colon definition (a program). Here, a word called ADD has

been defined which adds two and seven and prints the result:

```
: ADD 2 7 + . ;
```

By simply entering ADD, the number nine will be displayed. This is not a very good way to add two numbers, so like Basic, Forth can get the two numbers at run-time:

```
: ADD + . ;
```

To add two numbers the following is entered:

```
2 5 ADD
7 Ok
```

Notice the 'ok' which is the equivalent of Basic's Ready prompt.

Reverse Polish Notation

Forth requires that numbers to be operated upon should be on the stack prior to the operator. In other words, to perform a mathematical function, it is necessary to place the operand(s) before the operator(s), and not mix them as we humans do:

3 7 + is the equivalent of 3 + 7

We humans perform our calculations in infix notation, but Forth does it in postfix notation, known as reverse polish notation. Here are some algerbraic expressions in infix notation (normal) with their postfix (Forth) equivalents:

Normal	**Forth**
A+B-C	A B + C -
(+B)*C	A B + C *
A*B-C/d+E	A B * C D / - E +

It looks confusing at first, but it is surprising how easy it is to adapt.

The reason why postfix notation is used is because computers find it easier. As a result, this also contributes to Forth's speed.

Definitions

In Basic programs words are written, but in Forth words are defined. To inform the micro that you are defining a Forth word, two Forth words ':' and ';' are used together with a name for that word:

```
: name forth-words;
```

Once the definition is entered, it is placed within the dictionary and can be used just like any other Forth word. Other Forth words could use other definitions.

One of the great advantages that Forth has over conventional procedure librarys is that it is possible with many versions of Forth to inform the package that you wish to make your words part of the language



itself. Thus the next time you load Forth your own defined words will be part of the language!

Finally, the following Basic Forth speed comparison demonstrates the speed of Forth. The two programs are identical in operation, and perform the following:

Clear the text screen
Fill the colour map with the value 1, i.e. white.
POKE to the screen display 26 complete screenfuls.

The result is that a screenful of A's will be displayed, then B's and so on. In BASIC the test took 115 seconds to complete, and in (Melbourne) Forth it took 10.23 seconds!

```
10 PRINT "[CLS]"
20 FOR C=55296 TO 56296
30 POKE C,I
40 NEXT C
50 FOR L=1 TO 26
60 FOR I=1024 TO 2039
70 POKE I,L
80 NEXT I
90 NEXT L
```

```
: BM1 CLS
      56297 55296 DO
       1 I C!
      LOOP
      27 P DO
       2040 1024 DO
        J I C!
       LOOP
      LLOP ;
```

Reviews

It is very easy to implement Forth on a micro-especially one like the 64. As a result, there are at least seven implementations of the language available in the UK.

As you can see from the list, the prices of Forth packages differ. The features offered also vary, and it is a good idea to think about why you need the package. If, for example, you wish to expand an existing Forth, with a no-frills package, then Fig-Forth could be what you are looking for. On the other hand, White Lightning from Oasis Software offers many extensions. Another very good version is Forth+ from Melbourne House. With this package it is possible to produce your own stand-alone programs with ease.

Reviewed here are just four of the packages currently available.

Fig-Forth
Romik Software
Tape only — £14.95

Although this version is one of the cheapest, and offers no 'extras' at all, it is a great package!

This is because of the way in which the package has been implemented. Romik Forth offers the user a true fig-Forth environment without any additions of any kind. This has its advantages. First, it leaves memory free that would otherwise be used for extensions that the user may not want in the first place. And secondly, the

user can expand and customise the package without having to worry about workspace that may be used by extensions.

This said, I wouldn't recommend Romik Forth for the absolute beginner. Forth is unfriendly by nature, and Romik's version, an exact implementation, reflects this!

In use, the Romik Forth proved very fast, even compared with other Forths. The only slow aspect was the screen output. I believe the authors have made a CALL to the Basic/kernal ROMS and used the same routines as Basic for screen output. This is a pity, as it is possible to optimise this routine, thus speeding it up.

Being an accurate implementation of Fig-Forth, Romik Forth uses the same editor as most Forths – and a horrible editor it is too!

When Forth was designed, no method was specified for inputting source code. A US member of the Forth Interest Group (hence Fig) designed a crude but practical line editor, published in the user groups newsletter. The author suggested it should be used until someone designed a better one. Unfortunately, everyone (including the software houses) decided to use it, and it's still being copied in Forth implementations today!

Forth-64
Audiogenic
Cartridge only—£29.95

The standard editor used in Fig-Forth is not the only 'off-putter' about the language. SAVEing and LOADing to and from a backup device is also complicated.

However, in this case, there is a justified reason. Forth was designed to use dynamic storage meaning that memory is treated as the backup device. The idea behind this is to speed-up SAVEing and LOADing. Only when you physically wish to store or LOAD the data (at the start or end of an editing session perhaps) would you FLUSH the data to disk or tape. In other words, dynamic store is a go-between the memory and tape/disk.

But managing the Ramdisk is a tricky business in standard Forth. Audiogenics Forth-64 gets round this problem by offering easy to use Forth words for saving and loading program screens. Not only that, but the IEEE bus is fully supported, so it is possible to control (with some ease) any device bolted on the back of the 64.

Audiogenic's Forth is supplied on cartridge thus SAVEing the user the drudgery of LOADing from tape to disk. Not only that, but being on cartridge frees memory for more source screens which is quite important.

Numerous words cover tape/disk I/O and the IEEE bus, but there are no graphic or sound extensions on the 64 version. But it should be noted that the Vic-20 version does have a word for changing the border and screen colour as well as loading the sound registers.

The editor used in this version is based around the original, but makes good use of the Commodore function/cursor keys. What is a bit unusual is that the screen used for direct mode is a Forth editing screen. This means that when the end of the screen is reached, it is necessary to clear it before continuing. This is quite awkward, and takes away some of the immediacy of direct execution.

However, screen (as in editing) management is very well organised with a lot of the mundane tasks being done automatically. A screen is selected, and after the program source has been edited, it can be loaded into main memory and run. To actually save or load a number of source screens, a file is specified in the SSAVE/SLOAD command, and Forth will save or load the required number of blocks – simple!

White Lightning
Oasis Software
Tape—£19.95, disk—£29.95

White Lightning is quite simply the best value for money you can get. For the price, you not only get a good implementation of Fig-Forth, but you also get a graphics development system.

White Lightning is a fairly complex package that can be used to develop commercial programs. The package is Forth based and includes a Fig-Forth nucleus. Built around this is a graphics development shell called IDEAL which contains numerous sprite and sound words which are used in the Forth program. As well as this, it is possible to access Basic within a White Lightning or machine code. Once the program has been developed and debugged, it is possible to save it in a stand alone version that can be freely distributed.

Both disk and tape versions are available, but the 'serious' programmer should opt for the disk version as it is necessary to save and load sprite data as well as the Forth source if a graphics program is being developed.

As mentioned it is possible to include Basic statements within the Forth source. This will mean that the program will not run as fast as it would in Forth only, but it is a great help when starting off with the language.

FORTH+
Melbourne House
Tape—£14.95

Although only recently released, Forth+ has a long and interesting history. This Fig-Forth based package was initially developed for the ZX Spectrum by Abbersoft.

Forth+ is supplied on cassette which is recorded in the Pavlova format and takes under three minutes to load. Disk users will be relieved to hear that the package can be transferred to disk with just two simple commands. Once saved to disk, the package takes 40 seconds to load and uses the disk as the main backing device.

Supplied with the cassette is a (grotty) 36 page manual which covers the bare essentials of the package. There are a number of features within the package that are not documented in the manual, but I understand that Melbourne House will be releasing a full length manual in the near future.

Forth+ is based around Fig-Forth and offers a number of interesting extensions. First, the screen editor has been adapted to make use of the C64's editor. Source code can be entered into any of the nine screens in a free format, and an edit session can be aborted if desired. One point missing from the editor is the ability to delete carriage returns. This may not sound important, but when editing in a free format (such as the example program) this becomes essential.

Another interesting feature is the ability to produce stand-alone programs with the package. After your masterpiece is created, ZAP followed by the name of the word will save a machine code file which can be SYSed from Basic. However, the code that is saved is as large as the Forth itself (50 blocks), and probably contains most of the package. There are more elegant methods of achieving the same thing.

Screen modes included within the package are:

MCM Multi-colour mode
NCM Normal colour mode
ECBM Extended background colour mode
NBCM Normal background colour mode
HRG High-res mode
LRG Low-res mode

All the normal colour functions such as border, ink and background are supported, together with plot and draw. Sixteen sprite commands are included as well as 16 sound commands.

In use, Forth+ proved to be excellent. Even a semi-fatal crash could be re-covered from re-SYSing back into the main code. The pakcage is not overwhelmed with features, but the facilities which it does have are perfectly adequate.

Other Forth Packages

There are numerous versions of FORTH available, and here is a list of some:

Forth 64	Handic Software	£34.95
Forth 64	Hesware	£54.95
Tiny-Forth	Adamsoft	£12.95, £14.95

NUTS & BOLTS
OF THE
C/16

Nick Hampshire brings you a detailed look at the C-16 Operating System variables and memory map.

ANYONE WISHING TO DO EXTENSIVE machine code programming on the C-16 or Plus-4 will require two essential pieces of information.

The first, is a list of the operating system variables, their location and function. This is essential if fatal interaction with the system software is not to occur. It is also important to know where to place variables when using any of the system subroutines.

The second essential piece of information is a memory map. This contains the entry points to the operating system and Basic ROM routines. The use of such routines within a program can save a considerable amount of time and program space, an important consideration with the C-16.

I have included both these pieces of information in this article.

Unfortunately, Commodore has not previously produced any of this data and it has therefore been necessary to use the experience gained with previous Commodore machines to deduce the location and function of both variables and routines. All the locations discovered have been thoroughly checked and I have no reason to doubt that they are correct, I would however be very interested to hear from any readers with additional information.

The operating system variables are, in common with all Commodore machines, stored in the bottom area of memory extending from location $0000 up to the bottom of the screen colour memory at $0800, a total of 2K of memory. All the important variable locations are shown in Table 1. It should be noted that locations $00 and $01 are an I/O port located on the processor chip and cannot therefore be used for variable storage.

A memory map of the ROM based operating system and Basic routines is shown in Table 2. Many of these routines are usable to some degree, and some are extremely valuable. Most of the routines and variables are similar, or even identical, in function, if not in location, to those on the C64.

Readers wishing to gain an in depth knowledge of the routines, functions and operations are recommended to consult one of the advanced books on the 64. The *64 Revealed* series by Nick Hampshire, published by Collins is recommended, especially *Advanced Commodore 64 Basic Revealed*.

Table 1

```
Operating system variable storage.
==================================

0001      - Cassette control
      Bit 4 : Cassette read
          3 : Cassette motor (0=on)
          1 : Cassette write
000A      - 0=LOAD,1=VERIFY
000D      - Type: FF=string,00=numeric
000E      - Type: 80=integer,00=floating
000F      - DATA scan/LIST quote/memory flag
0014-0015 - Integer value
0016      - Pointer:temporary string stack
0017-0018 - Last temp string vector
0019-0021 - Temporary string stack
0022-0025 - Utility pointer area
0026-002A - Prooduct area for multiplication
002B-002C - Pointer:start of Basic
002D-002E - Pointer:start of Basic variables
002F-0030 - Pointer:start of arrays
0031-0032 - Pointer:end of arrays
0033-0034 - Pointer:bottom of strings
0035-0036 - Pointer:current string
0037-0038 - Pointer:top of Basic memory
0039-003A - Current Basic line number
003B-003C - CHARGET pointer
003D-003E - Pointer:FOR/DO stack
003F-0040 - Current DATA line number
0041-0042 - Current DATA address
0043-0044 - Input vector
0045-0046 - Current variable name
0047-0048 - Current variable address
0049-004A - Variable pointer for FOR/NEXT
004B-004C - Y save;op save;Basic ppointer save
004D      - Comparison symbol accumulator
004E-0053 - Misc numeric work area
0054-0056 - Jump vector for functions
0057-0060 - Misc work area
0061      - FAC#1 exponent
0062-0065 - FAC#1 mantissa
0066      - FAC#1 sign
0067      - Series evaluation constant pointer
```

Table 1(continued)

0068	- FAC#1 overflow
0069-006E	- FAC#2
006F	- FAC sign comparison
0070	- FAC#1 rounding
0075	- Room for graphics screen (0=not available)
007C-007D	- Pointer:GOSUB stack
0083	- Flag for window ($40=window on, $80=multicolour,$C0=both
0090	- Status word ST
0091	- Keyswitch CIA:STOP and RVS flags
0093	- 0=load,1=verify
0094	- Serial output:deferred char flag
0095	- Serial deferred character
0097	- # open files
0098	- Input device
0099	- Output CMD device
009A	- Direct=80/run=0 output control
009D-009E	- Tape end address/End of program
00A3-00A5	- Jiffy clock
00A6	- Serial bit count/EOI flag
00AA	- Countdown tape write/bit count
00AB	- # chars in filename
00AC	- Current logical file
00AD	- Current secondary address
00AE	- Current device
00AF-00B0	- Pointer to filename
00B2-00B3	- I/O start address
00B4-00B5	- Alt start address (load/verify)
00B6-00B7	- Pointer:cassette buffer
00C4-00C5	- Input cursor log (row,column)
00C6	- Which key:64=no key
00C7	- Input from screen/keyboard
00C8-00C9	- Pointer to screen line
00CA	- Pointer:cursor column
00CB	- Output quotes flag
00CD	- Pointer:cursor row
00CE	- Output character (to screen)
00CF	- # of inserts outstanding
00EA-00EB	- Screen colour pointer
00EC-00ED	- Keyboard pointer
00EF	- # chars in keyboard buffer
00F8	- Type of tape file
00FF-010A	- Floating to ASCII work area
0100-013E	- Tape error log
0100-01FF	- Processor stack area
0200-0258	- Basic input buffer
0259-025A	- Pointer: line# for CONT
025B-025C	- Pointer:Basic statement for CONT
02F2-02F3	- Float-fixed vector
02F4-02F5	- Fixed-float vector
0300-0311	- Basic vectors

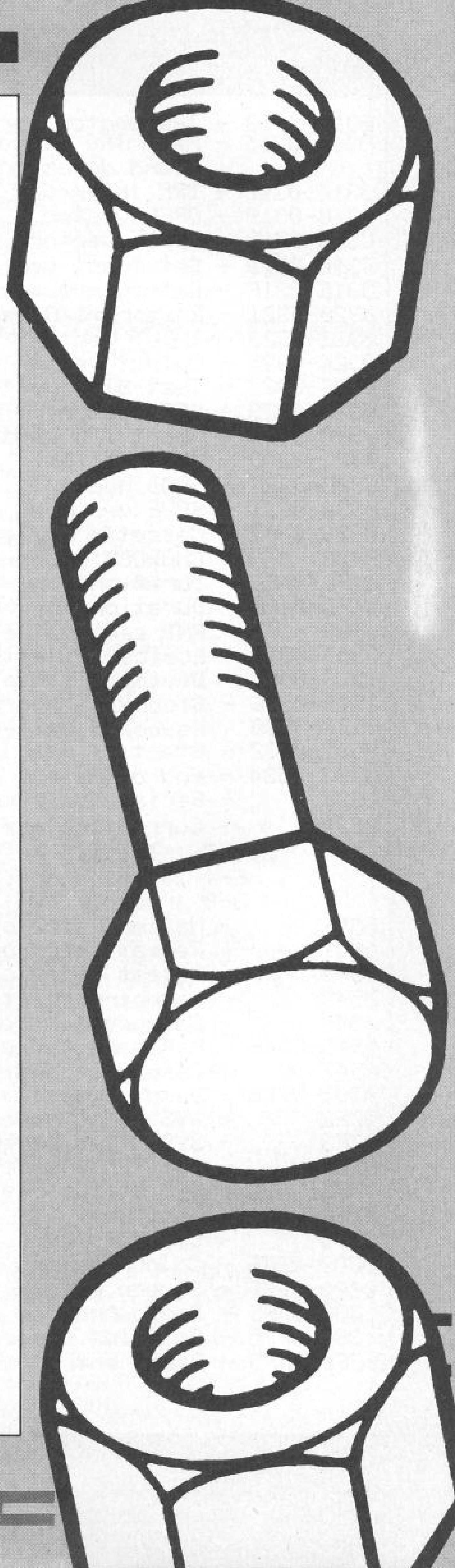


Table 1 (continued)

```
0312-0313 - IRQ vector for keyscan/clock
0314-0315 - Main IRQ vector for
            sound duration/graphics split
0316-0317 - BRK interrupt vector
0318-0319 - OPEN vector
031A-031B - CLOSE vector
031C-031D - Set input vector
031E-031F - Set output vector
0320-0321 - Restore I/O vector
0322-0323 - INPUT vector
0324-0325 - Output vector
0326-0327 - Test-STOP vector
0328-0329 - GET vector
032A-032B - Abort I/O vector
032C-032D - User vector
032E-032F - LOAD vector
0330-0331 - SAVE vector
0333-03F2 - Cassette buffer
0473      - CHARGET subroutine
04FC/04FE - Duration for voice 1
04FD/04FF - Duration for voice 2/noise
0503      - RND seed value
0509-0512 - Logical file table
0513-051C - Device # table
051D-0526 - Secondary address table
0527-0530 - Keyboard buffer
0531-0532 - Start of usable memory
0533-0534 - End of usable memory
0535      - Serial bus timeout flag
053B      - Current colour code
      Bit 7 : 1=flash
        6-4 : luminance (0-7)
        3-0 : colour (0-15)
053F      - Maximum size of keyboard buffer
0541      - Repeat speed counter
0542      - Repeat delay counter
0543      - Keyboard shift/control flag
0544      - Case switch count
0545-0546 - Keyboard table setup pointer
0547      - Case switch disable
0700-07B0 - Gosub stack
07F2      - SYS A reg save
07F3      - SYS X reg save
07F4      - SYS Y reg save
07F5      - SYS status reg save
07F6      - Last key
07FD      - Countdown for double TI bump
0800-0BFF - Colour memory
0C00-0FFF - Screen memory
1000-3FF5 - Basic program memory
1800-3FF5 - Graphics screen/colour memory
3FF6-3FFD - Reset entry (when ROM is out)
```

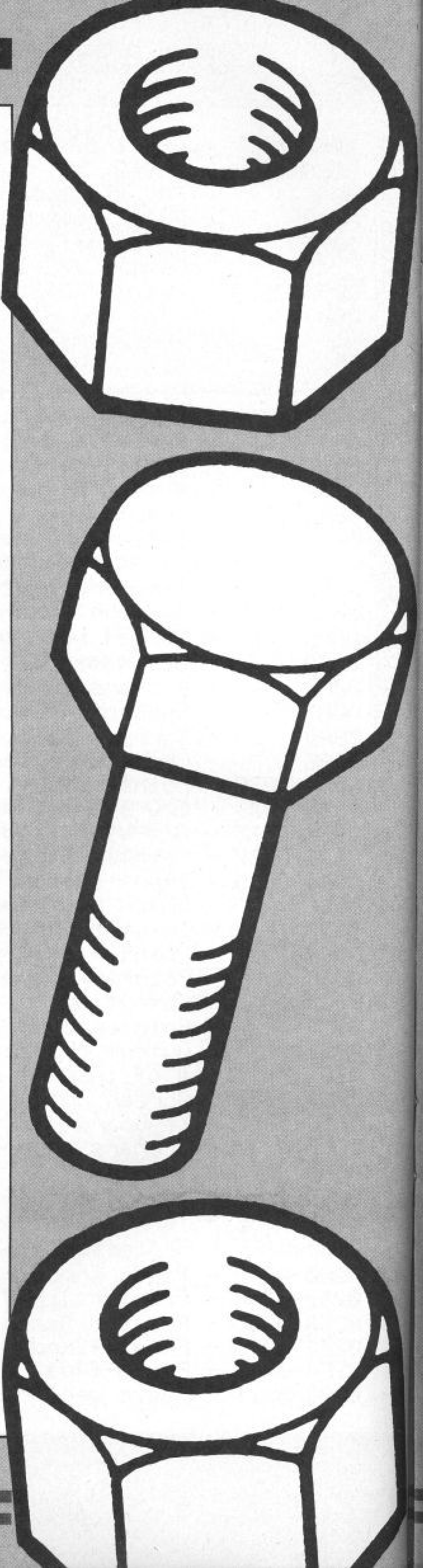


Table 2

Commodore 16 Memory Map.
========================

8000 - Basic ROM cold start routine
8019 - Initialize Basic
80CC - Power-up message
8105 - vectors for $0300
8117 - Initialize vectors
8123 - CHRGET for $0479
818E - Keywords
8383 - Command vectors
8415 - Function vectors
8453 - Arithmetic operator vectors
8471 - Error messages
8681 - 'SYNTAX ERROR'
8683 - Error routine
8703 - 'READY.'
870F - Basic warm start
8A3D - Find Basic line
8A79 - Perform 'NEW'
8A98 - Perform 'CLR'
8AF1 - Set text pointer
8AFF - Perform 'LIST'
8BBC - Perform 'RUN'
8C9A - Perform 'RESTORE'
8CD8 - Perform 'STOP'
8CDA - Perform 'END'
8D03 - Perform 'CONT'
8D2C - Perform 'GOSUB'
8D4D - Perform 'GOTO'
8D83 - Perform 'RETURN'
8DB0 - Perform 'DATA'
8DE1 - Perform 'IF'
8E0B - Perform 'REM' and 'ELSE'
8E1B - Perform 'ON'
8E3E - Get line number (0-63999)
8E7C - Perform 'LET'
8FE0 - Perform 'PRINT#'
8FE6 - Perform 'CMD'
9000 - Perform 'PRINT'
9088 - Print string from (y.a)
90B8 - Perform 'GET'
90EE - Perform 'INPUT#'
9108 - Perform 'INPUT'
914F - Perform 'READ'
920B - '?REDO FROM START' message
9294 - Perform 'NEXT'
932C - Evaluate expression
9439 - Constant - pi
9465 - Perform 'NOT'
948B - Scan past ')'
948E - Scan past '('
9491 - Scan past ','
95F8 - Perform 'OR'
95FB - Perform 'AND'
9628 - Perform comparisons (<=>)
969B - Perform 'DIM'
96A5 - Locate variable
986C - Constant - 32768
9886 - Float-fixed
9A62 - Perform 'FRE'
9A7D - Perform 'POS'
9A9D - Perform 'DEF'
9ADE - Perform 'FN'
9B66 - Perform 'STR$'
9C48 - Get temporary string
9CBB - Perform 'CHR$'
9CCF - Perform 'LEFT$'
9D03 - Perform 'RIGHT$'
9D15 - Perform 'MID$'
9D61 - Perform 'LEN'
9D70 - Perform 'ASC'
9D84 - Get 1 byte parameter
9D93 - Perform 'VAL'
9DD2 - Get parameters for POKE/WAIT
9DFA - Perform 'PEEK'
9E12 - Perform 'POKE'
9E6A - Perform 'WAIT'
9E87 - Perform 'subtract'
9E9E - Perform 'add'
A01E - Perform 'LOG'
A07B - Perform 'multiply'
A197 - Perform 'divide'
A221 - Memory to FAC#1
A281 - FAC#2 to FAC#1
A2A0 - Round FAC#1
A2BE - Perform 'SGN'
A2DD - Perform 'ABS'
A358 - Perform 'INT'
A5E4 - Perform 'SQR'
A5EE - Perform 'power'
A627 - Perform 'negate'
A660 - Perform 'EXP'
A707 - Perform 'RND'
A77D - Basic I/O error handler
A785 - Basic-kernal patch for OPEN
A78B - Basic-kernal patch for PRINT
A791 - Basic-kernal patch for INPUT
A797 - Basic-kernal patch for set output device
A7A6 - Basic-kernal patch for set input device
A7AF - Basic-kernal patch for GET
A7B5 - Perform 'SYS'
A7DE - Perform 'SAVE'
A7F0 - Perform 'VERIFY'
A7F3 - Perform 'LOAD'
A84D - Perform 'OPEN'
A85A - Perform 'CLOSE'
A86B - Get parameters for LOAD/SAVE
A8B0 - Get parameters for OPEN/CLOSE
A954 - Garbage collect
AA70 - Perform 'COS'
AA77 - Perform 'SIN'
AAC0 - Perform 'TAN'
AB1A - Perform 'ATN'

Table 2 (continued)

```
AB8D - Perform 'RENUMBER'
ADCA - Perform 'FOR'
AE5A - Perform 'DELETE'
AECA - Get range for LIST/DELETE
B42B - Perform 'TRAP'
B440 - Perform 'RESUME'
B544 - Perform 'PUDEF'
B557 - Perform 'DO'
B5AC - Perform 'EXIT'
B603 - Perform 'LOOP'
B652 - Perform 'TRON'
B655 - Perform 'TROF'
B6CD - Perform 'AUTO'
B6E8 - Perform 'HELP'
B729 - Perform 'KEY'
B849 - Perform 'SOUND'
B8BD - Perform 'VOL'
B8D1 - Perform 'PAINT'
B9D4 - Perform 'CHAR'
BAE2 - Perform 'BOX'
BD35 - Perform 'GSHAPE'
BE29 - Perform 'SSHAPE'
C01E - Perform 'CIRCLE'
C4D9 - Perform 'DRAW'
C50D - Perform 'LOCATE'
C51A - Perform 'COLOR'
C567 - Perform 'SCNCLR'
C5B8 - Perform 'SCALE'
C5C3 - Perform 'GRAPHIC'
C8BC - Perform 'DIRECTORY'
C941 - Perform 'DSAVE'
C951 - Perform 'DLOAD'
C968 - Perform 'HEADER'
C99C - Perform 'SCRATCH'
C9CC - Perform 'COLLECT'
C9DA - Perform 'COPY'
C9F4 - Perform 'RENAME'
CA00 - Perform 'BACKUP'
CB21 - Get paramters for disk commands
CCCF - Read disk error
CD32 - 'ARE YOU SURE?' message
------------------------------------
CE00 - IRQ/BRK entry
CE0E - IRQ routine ($0314) - handles
       graphics split, sound duration
CE42 - IRQ routine ($0312) - handles
       clock, keyboard
CECD - Handle sound durations
CEF0 - Bump clock
CF26 - Get time
CF2D - Set time
D000 - Character definitions (2K)
D802 - Screen address low
D81B - Screen address high
D834 - Get screen size
D839 - Put/get row/column
D8A8 - Set screen pointers
D8C1 - Remove char from keyboard
D965 - Input until carriage return
DB11 - Read keyboard
DC49 - Output to screen
E01E - Keyboard select vectors
E026 - Unshifted table
E067 - Shifted table
E0A8 - CBM key table
E0E9 - Control table
E12A - Shift/run equivalent
E2EA - Delay for 0.5 sec
E9CC - Find any tape header
EA21 - Find a specific header
EBD9 - Get ($0328)
EBE8 - Input ($0322)
EC4B - Output ($0324)
EC8B - Receive from serial
ECDF - Send serial deferred
ED18 - Set input device ($031C)
ED60 - Set output device ($031E)
EDFA - Send 'talk'
EE1A - Send talk SA
EE2C - Send 'listen'
EE4D - Send listen SA
EE5D - Close file ($031A)
EEE8 - Find file entry
EEF8 - Get file details
EF08 - Abort all files ($032A)
EF0C - Restore default I/O ($0320)
EF23 - Send 'unlisten'
EF3B - Send 'untalk'
EF53 - Open file ($0318)
F04A - Load program ($032E)
F1A4 - Save program ($0330)
F265 - Test STOP key ($0326)
F2A4 - Power reset entry
F2CE - Set kernal vectors
F2D3 - Store kernal vectors
F30B - Initialise I/O
F352 - Initialise system constants
F40C - Set filename details
F413 - Set file details
F41A - Flag status
F41C - Get status
F423 - Set timeout
F427 - Read/set top of memory
F429 - Read top of memory
F42F - Set top of memory
F436 - Read/set bottom of memory
F438 - Read bottom of memory
F43E - Set bottom of memory
F445 - Monitor call entry
F44C - Monitor BRK entry ($0316)
       User vector ($032C)
FC19 - Get I/O address
FCB3 - IRQ entry
FCBE - IRQ exit
FD00 - TED memory
FF52 - Perform 'MONITOR'
```

This month A.P. and D.J. Stephenson provide the key to Basic and devle into the mysteries of keywords.

THE BASIC FACTS

SOME READERS WHO HAVE BEEN following this series may have found the last few articles a little tough. This month, we thought it would be nice to pause for breath and go over some of the lesser used BASIC keywords in detail.

The Basic language, as implemented in the C64, employs a variety of keywords which are sufficient to cope with most situations. A good proportion of the total are in constant use and have been treated and used in earlier articles in this series. For various reasons we have neglected some of them altogether. A few others have been used without adequate description. Although all keywords are defined in the user manuals, alternative treatments can often lead to a better understanding.

The vocabulary of any language is seldom utilised to the full. Indeed, in everyday speech, we only use a fraction of the total number of words we know. It is the same with programmers. However rich the Basic vocabulary, it is easy to get into a rut by sticking only to those keywords which are easy to understand or use. Unfortunately, the most ardent admirer of the C64 or Vic 20 would be forced to admit that the Basic vocabulary in these machines is, to say the least, sparse. Because of this, it is particularly important that we know how to make efficient use of every keyword. Before dealing with them individually, it pays to classify them into order.

Types of keyword

A keyword is any combination of characters, chosen for their mnemonic value, recognised by the interpreter as an order to be carried out. They fall into one of three classes, – statements or functions.

Commands are keywords which have an overall effect on the complete program. They are more often employed outside a program in direct mode. However, they will also work within a program under a line number. The following keywords are direct commands: CONT, LIST, LOAD, NEW, RUN, SAVE, VERIFY.

Statements are keywords which perform some particular action within a program. Most keywords are statements.

Functions are specialised statements which perform a standardised operation on a variable. Functions can be recognised by the brackets which enclose the variables. The following complete list of Commodore functions uses X or Y as example numberic variables and A$ for string variables: ATN(X), CHR$(X), COS(X), EXP(X), FRE(X), INT(X), LEFT$(A$,X), LEN(A$), LOG(X), MID$(A$X,Y), PEEK(X), POS(X), RIGHT$(A$X), RND(X), SIN(X), SPC(X), SQR(X), STR(X), TAB(X), TAN(X), USR(X), VAL(A$).

TI and TI$ are exceptional in that Commodore classifies them as functions although they do not require brackets round the variable. Numeric variables used in functions are often called arguments. The arguments in two of the functions, POS and FRE, are dummies, meaning that some number must be entered to avoid triggering a syntax error although it doesn't matter what number you chose within the range 0 to 255.

The command RUN

RUN clears all variables when it starts a program from the first line number. It is possible to start a program from lower down the program by using RUN X, where X is the starting line number. RUN X will often lead to error conditions because variables created under previous line numbers are cleared from memory.

If you want to restart the program a second time at some lower point, it is better to use GOTO X which leaves previously created variables unharmed. The main danger with using GOTO X is triggering a 'REDIM'D ARRAY' error if a DIM line is re-encountered.

The command CONT

A program comes to a halt under any one of the following conditions:

(a) The RUN/STOP key has been pressed.
(b) The program has reached either a STOP or END statement.
(c) The program has reached the last line number.
(d) The program has encountered a standard error condition.

The program can always be restarted from the beginning or, by typing RUN X, started from line X. However, RUN or RUN X always clears variables from memory which were created by the last run. Only the program is retained.

The command CONT can provide a powerful weapon for sniffing out bugs in a program during the development stage. It is usually possible, providing the development has proceeded in accordance with guidelines of good structure, to recognise certain critical points in a program.

To check the value of variables at these critical points, temporary STOP instructions can be inserted. When the program is RUN, it will halt at the first STOP and you can check the state of the variables by printing out their value. If the values are different to what you expected there is little point in proceeding with the rest until the reason for the discrepancy has been found. Assuming that everything is OK, you can allow the program to carry on with the next segment by again using CONT.

Unlike RUN, which always starts the program afresh, CONT allows it to carry on from where it was stopped and preserves the value of all variables reached up to that point. These temporary STOPs act as 'break points' in the program, enabling the status to be investigated and faults remedied in a series of easy stages.

To take full advantage of break points, the program should use fixed, easy to use, values so the results of calculations can easily be checked manually at the end of each one. Remember that CONT will not

work if the program has halted due to an error condition. If you try it, the response will be 'CAN'T CONTINUE.

The command NEW

As far as Basic is concerned, the command NEW will clear the entire memory, including any pre-existing program. Although NEW is usually entered in direct mode, you may include it within the program under a line number. If you start on a new program when the previous one or its remnants are stil in memory, be prepared for some unexpected results. Switching OFF and ON again will clear everying but entering NEW is more elegant.

Authorities disagree on the wisdom of repeatedly switching the power on and off more than absolutely necessary. Repeated use of the main switch may even reduce the life of some of the more sensitive chips.

The command LIST

This is probably the most overworked command in Basic. Although the default listing is to the screen, a previously active CMD statement can be used to output the listing to a printer or other suitable peripheral.

You may include LIST within a program under a line number but, after the listing, the program will always revert to command mode and the READY message. Unless it is placed at the line immediately preceding the END statement, it is difficult to imagine what purpose LIST can serve within a program other than saving you the trouble of entering LIST afterwards. But then you don't always want a listing after every run!

The command SAVE

This command can only be used to save programs. It saves on tape by default or on disc by quoting device number eight after the program name. The Commodore method of saving on tape is slower than some other machines, partly because two copies are always saved so that read errors on playback are easily pinpointed by discrepancies between the versions.

If SAVE is used within a program under a line number, the program is halted while the tape or disk is being recorded. On completion of the save operation, it carries on with the rest of the program. In this way, it is possible to save a program and run it in one go, although you'll probably never need to do this.

The LOAD command

When you load a program using LOAD as a direct command, all variables are cleared from memory including any

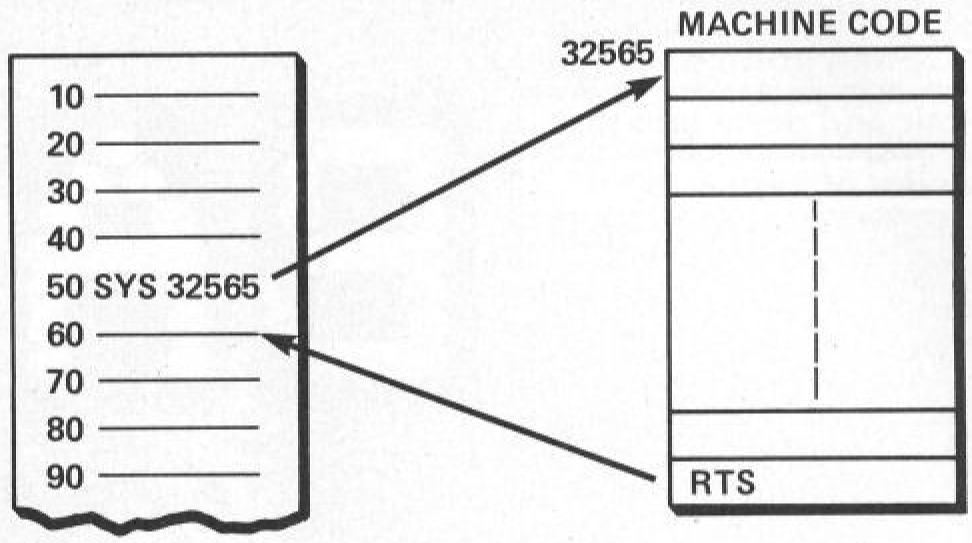


Figure 13.1

resident program which may be there. If LOAD is used as a statement within a program, although the existing program is cleared, the existing variables are untouched and a RUN automatically follows. Used in this way, the LOAD command acts as a 'chain' command so that several programs on tape or disk can be chained together. This allows large programs which would exceed the bounds of memory, to be entered in parts small enough to be accommodated within the memory space available. Each subsequent program finds an environment already prepared by previous programs. The individual smaller programs must be carefully designed to ensure that, apart from their use of existing variables, they are self sufficient.

For example, one program cannot GOTO a line number of another because it would not be in the computer's memory. The program bytes are normally loaded into a block of memory starting at address 2048 (0800 hex) onwards.

The VERIFY command

Pessimists use VERIFY immediately after they have saved a program on tape. It takes a long while to save a large program and just as long to verify it afterwards so it is a precaution which many programmers neglect to take. In any case, the most likely reason for failing to LOAD is an error during playback rather than during record.

However, VERIFY does have one additional advantage. Because it leaves the tape several inches after the program's end it is safe to receive another program immediately without worrying about overlap.

POKE and PEEK

These two statements are best considered together because they occupy a grey area, midway between Basic and the inner workings of the machine. In traditional Basic, the individual pigeon holes in memory are of no consequence because the interpreter takes care of all memory assignments.

PEEK(X) enables us to have a look at the byte currently occupying memory address X. To avoid an 'ILLEGAL QUANTITY' message, X must lie within the range 0 to 65,535. (There is no memory outside this range).

POKE X,Y allows us to store the number Y in address X. Since a memory location can only hold one byte of data, Y must be within the range 0 to 255, and it must be a positive number.

To PEEK a number can never be harmful but an indiscriminate POKE can turn sour. Poking the right number into the wrong address or vice versa crash a program. The results of a crash vary but it's usually either a screenful of dazzling screen characters or a stoney indifference to all keyboard activity. The crash will almost certainly occur if a number is poked into one of the workspace areas used by the operating system.

You should regard the operating system with respect as it's capable of withdrawing all co-operation if crossed.

Providing care is taken in choosing memory addresses, POKE can be used to:

(a) Assign date bytes directly. For example, if we want to store numbers less than 256 in memory, it can be done far more economically by using POKE than by assigning the number to a Basic variable. It is not so convenient or flexible as normal assignments but worth bearing in mind when a lot of small numbers have to be manipulated.
(b) As one of the elements in graphic work.
(c) Passing parameters to machine code surroutines called from within Basic.
(d) In spite of the warning regarding a POKE into operating system areas, there are a number of special

locations which can be POKED in order to modify certain effects.

(e) To control colours. POKE 53282,8 will set background colour to orange.

(f) It is possible to operate on selected bits of a memory byte by using the AND or OR operators in conjunction with POKE and PEEK. For example, bit 4 in location 53270 decides whether the multi-colour character mode is on (bit 4=1) or off (bit 4=0). To ensure the off condition, use

```
POKE 53270,PEEK(53270) AND 239
```

This may seem a trifle obscure unless you are familiar with the bitwise features of AND.

The AND statement

It should not be difficult to understand the meaning of AND and OR when used in conjunction with the IF/THEN structure. For example,

```
100 IF A<B AND C<D THEN PRINT X$
```

The syntax is almost self explanatory and clearly means that both conditions must be true. However, there are other, less obvious, qualities lurking beneath the surface which are known as bitwise operations. Study the following series of operations:

Assume this pattern	11011011
Now AND it with	11101111
The result is	11001011

Note that the result is the same as the original pattern except where the AND pattern had a 0 in that position. The AND pattern, known as a mask, is calculated thus: Use '0' in the mask when you want '0's in the result, otherwise use '1's.

Let's try a practical example. Suppose a certain memory location holds the following bit pattern:

10011100

Let's also suppose that we wish to clear bit 4 or 0 without disturbing the others. (Bits are numbered 0 to 7, from right to left so bit 4 is the fifth bit from the right). The correct AND mask will be 11101111. Unfortunately, there is no provision in C64 Basic for entering bit patterns direct. We have to use decimal so we must convert the bit patterns in the AND mask to decimal before it can be accepted. The above example required an AND mask of 11101111.

Working laboriously in decimal, this becomes 1+2+4+8+323+64+128 = 239. Pity Commodore doesn't provide hex numbers – this would make AND mask calculations dead easy.

We are now in a position to return to the problem we left in the last section. You will remember that the following line was supposed to clear bit 4 in location 43270 to 0:

```
POKE 53270,PEEK(53270) AND 239
```

We are poking the same bit pattern back into the location after we have ANDed it with 239. We know that 239 decimal is 11101111 which is the correct AND mask for clearing bit 4 in the original location. To consolidate, confirm that the following apparent absurdities are in fact quite true: 247 AND 251 = 251, 7 AND 8 = 0.

The OR statement

This, like the AND, has bitwise connotations. As we have seen, ANDing is used when we want to clear certain bits to 0 but ORing is used when we want to set certain bits to 1. The rule for the OR mask is a follows: Use '1's in the mask where bits are to be '1's, otherwise use '0's.

Suppose we start with	11001011
Then we OR it with	00010000
The result is	11011011

Note that the result is the same as the original except in the position where there was a 1 in the OR mask.

Let's return to our previous example once again but this time, assume we want bit 4 in location 53270 to be a 1. The OR mask must be 00010000, which in decimal is 16. Our POKE line then becomes:

```
POKE 53270,(PEEK 53270) OR 16
```

To consolidate, confirm the following: 7 OR 7 = 7, 5 OR 2 = 7.

The NOT statement

This is the third member of the trio capable of bitwise operations. Before describing its action, we must first define a few terms: To flip, means to change a 1 to 0 or vice versa. The bit complement (logical complement or one's complement) is the bit pattern formed by flipping all the bits. The two's complement is the same as the bits complement except that an extra 1 is added. Example:

Starting with	10010111
Bit complement is	01101000
Two's complement is	01101001

The NOT statement forms the two's complement of a bit pattern which can lead to rather mysterious results. For example, NOT 1 = 255. To see why, note that 1 = 00000001 so the two's complement is 11111110 + 1 = 255.

The mystery clears up when we learn that a negative number, as far as the computer is concerned, is really the two's complement form of the equivalent positive. When we NOT something, we are really asking for the negative version.

All this information on the bitwise operations of AND, OR and NOT is really crossing the border between Basic and machine code. The same thing can be said regarding the next item.

The SYS command

It is possible to mix Basic with machine code in the one program. There are several situations where such a mix may be justified. The execution speed of Basic is sufficient for some purposes but intolerable for others. Another disadvantage of Basic is that memory is not used economically. For example, in Basic, the memory set aside for numbers is fixed, irrespective of their magnitude. It takes just as many bytes to store the number three as it does to store 3000. Machine code programs can arrange for memory storage to be more closely tailored to magnitude.

However, machine code is not everyone's idea of the good life so, as far as possible, most of the program may continue to be written in the tranquil environment of Basic, with only the occasional leap into and out of machine code. Figure 13.1 shows how such leaps can be performed using the keyword SYS.

Note from the figure that we can only jump to a machine code subroutine if we happen to know where it is located in the memory map. In other words, we must know the starting address. If you've written the machine code there's no problem but you may want to use some of the ready made versions already residing in the ROM. These are free, providing you find out the starting address by consulting the Programmer's Reference Guide or employing some crafty detective work. Remember you can always PEEK locations in ROM.

There is one obstacle that can arise in the use of SYS. Many subroutines only work if information is provided. To quote a simple example, a subroutine to find the square of a number obviously can't work unless it is supplied with the number to be squared. When such a subroutine is written, it will assume the number to be already residing in a particular memory location. The address will form part of the accompanying documentation.

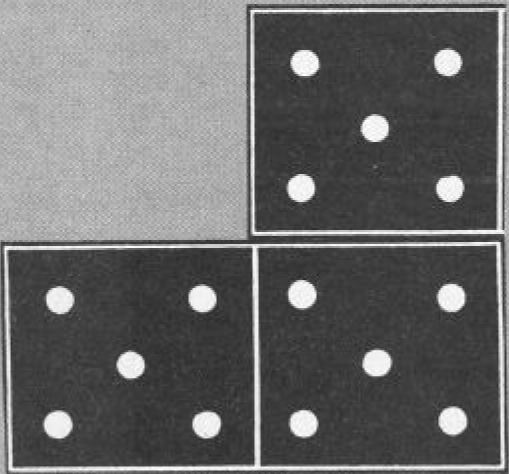
Therefore, before we call on the subroutine by means of SYS, we must poke this information in the required address. Thus a typical call might look like this:

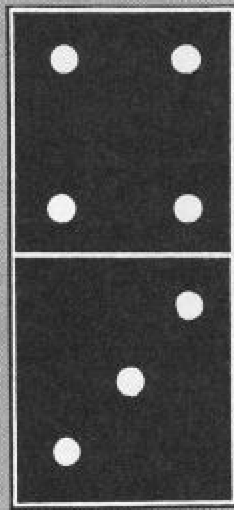
```
100 POKE 34520,16
110 SYS 34500
```

The data location is 34520 and the number 16 has been poked into it. The subroutine is assumed to start at address 34500. The machine code subroutine must end with RTS (Return from Subroutine) otherwise control will not come back to Basic.

SPOTS BEFORE YOUR EYES

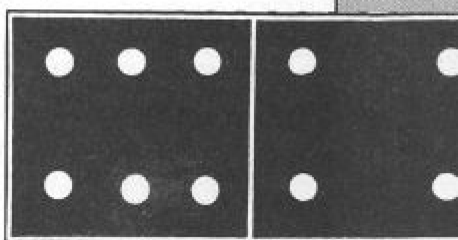
Take on your 64 at dominoes with this listing from Roy Titchard.



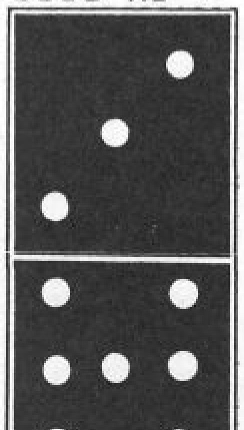


```
100 REM        DOMINOES
110 REM BY ROY TITCHARD 1985
120 GOSUB5930
130 :
140 FORNM=1TO15:READMN:NM$=NM$+CHR
$(MN):NEXTNM
150 DATA66,89,32,82,79,89,32,84,73
,84,67,72,65,82,68
160 POKE53280,0:POKE53281,0
170 :
180 PRINT"[CLEAR][DOWN][DOWN][DOWN
]":FORA=0TO3:PRINTTAB(10)DO$(A);:N
EXT
190 :
200 PRINT"[DOWN][DOWN][DOWN][DOWN]
              ";NM$
210 :
220 PRINT"[DOWN][DOWN][DOWN][DOWN]
[DOWN][DOWN][YELLOW]    [RVSON] DO
 YOU NEED IN
STRUCTIONS [RVSOFF] [Y-N]"
230 GETA$:IFA$=""THEN230
240 IFA$="N"THEN1210
250 IFA$="Y"THEN280
260 GOTO230
270 :
280 PRINT"[CLEAR][GREEN]
[RVSON] INSTRUCTIONS  [RVSOFF]"
290 :
300 PRINT"[DOWN][CYAN]NORMAL DOMIN
O RULES APPLY..........."
310 :
320 PRINT"[DOWN]YOU WILL BE PLAYIN
G WITH A DOUBLE SIX"
330 :
340 PRINT"[DOWN]PACK AND DEALT NIN
E DOMINOES."
350 :
360 PRINT"[DOWN]THE FIRST 'DROP' W
ILL BE RANDOM EITHER"
370 :
380 PRINT"[DOWN]YOU OR THE COMPUTE
R GOING FIRST."
390 :
400 PRINT"[DOWN]IF YOU GAIN 1ST DR
OP THEN YOU CHOOSE"
410 :
420 PRINT"[DOWN]YOUR DOMINO BY NUM
BER [1 TO 9]"
430 :
440 PRINT"[DOWN]THE DOMINO IS THEN
 DISPLAYED IN THE"
450 :
460 PRINT"[DOWN]CENTRE OF THE SCRE
EN."
470 :
480 PRINT"[DOWN][DOWN]        [YELLO
W][RVSON] PRESS RETURN TO CONTINUE
 [RVSOFF]"
490 GETA$:IFA$=""THEN490
500 IFA$<>CHR$(13)THEN490
510 :
520 PRINT"[CLEAR][DOWN]THE COMPUTE
R WILL AUTOMATICALLY MAKE"
530 :
540 PRINT"[DOWN]ITS MOVE AND RECOR
D THE MOVE AT THE "
550 :
560 PRINT"[DOWN]TOP RIGHT OF THE S
CREEN."
570 :
```

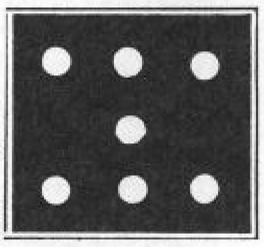
```
580 PRINT"[DOWN]THE CENTRE DOMINO
WILL THEN CHANGE TO"
590 :
600 PRINT"[DOWN]SHOW THE MOVE PLAY
ED (I.E) YOU DROP"
610 :
620 PRINT"[DOWN]DOUBLE BLANK THE C
OMPUTER PLAYS ONE"
630 :
640 PRINT"[DOWN]BLANK SO THE CENTR
E DOMINO CHANGES TO"
650 :
660 PRINT"[DOWN]BLANK ONE, AND SO
ON."
670 :
680 PRINT"[DOWN]A RECORD OF YOUR M
OVES IS KEPT ON THE"
690 :
700 PRINT"[DOWN]BOTTOM RIGHT OF TH
E SCREEN."
710 :
720 PRINT"[DOWN]     [WHITE][RVSON]
 PRESS RETURN TO CONTINUE [RVSOFF]
"
730 GETA$:IFA$=""THEN730
740 IFA$<>CHR$(13)THEN730
750 :
760 PRINT"[CLEAR][DOWN][c 1]IF YOU
R MOVE COULD ALTER THE STATE OF"
770 :
780 PRINT"[DOWN]PLAY (I.E) CENTRE
DOMINO READS ONE-TWO"
790 :
800 PRINT"[DOWN]AND YOU PLAY ONE-T
WO THEN YOU WILL BE"
810 :
820 PRINT"[DOWN]OFFERED THE CHOICE
 OF MAKING IT EITHER"
830 :
840 PRINT"[DOWN]ONE UP OR TWO UP (
BOTH ENDS THE SAME)"
850 :
860 PRINT"[DOWN]YOU THEN PRESS FRO
M '0'(BLANK) TO 6"
870 :
880 PRINT"[DOWN]DEPENDING ON WHICH
 DOMINO YOU CHOOSE."
890 :
900 PRINT"[DOWN][WHITE][RVSON] IF
YOU CANNOT PLAY A DOMINO THEN [RVS
OFF]"
910 :
920 PRINT"[DOWN]           [RVSON] P
RESS RETURN [RVSOFF][c 1]"
930 :
940 PRINT"[DOWN]YOUR MOVE WILL THE
N BE RECORDED AS [YELLOW][RVSON]OU
T[RVSOFF][c1]"
950 :
960 :
970 PRINT"[DOWN][DOWN]       [CYAN]
[RVSON] PRESS RETURN TO CONTINUE [
RVSOFF]"
980 GETA$:IFA$=""THEN980
990 IFA$<>CHR$(13)THEN980
1000 :
1010 PRINT"[CLEAR][DOWN][DOWN][YEL
LOW]IF A GAME IS 'STITCHED' (NEITH
ER PLAYER"
1020 :
1030 PRINT"[DOWN]CAN GO) THEN THE
WINNER IS THE PLAYER"
1040 :
1050 PRINT"[DOWN]WITH THE LEAST NU
MBER OF SPOTS."
1060 :
1070 PRINT"[DOWN][DOWN]THE DOMINOE
S ARE COLOUR CO-ORDINATED"
1080 :
1090 PRINT"[DOWN]FOR EASY IDENTIFI
CATION."
1100 :
1110 PRINT"[DOWN][DOWN][RED]    ON
E[YELLOW] IS RED : TWO IS YELLOW "
1120 :
1130 PRINT"[DOWN][GREEN]    THREE[
YELLOW] IS GREEN :[CYAN]FOUR[YELLO
W] IS CYAN"
1140 :
1150 PRINT"[DOWN][PURPLE]    FIVE[
YELLOW] IS PURPLE :[c 1]SIX[YELLOW
] IS ORANGE"
1160 :
1170 PRINT"[DOWN][DOWN]     [c 6][
RVSON] PRESS RETURN FOR FIRST GAME
 [RVSOFF]"
1180 GETA$:IFA$=""THEN1180
1190 IFA$<>CHR$(13)THEN1180
1200 :
1210 PRINT"[CLEAR]":GOSUB5930
1220 GU$(0)="BLANK "
1230 GU$(1)="ONE   "
1240 GU$(2)="TWO   "
1250 GU$(3)="THREE "
1260 GU$(4)="FOUR  "
1270 GU$(5)="FIVE  "
1280 GU$(6)="SIX   "
1290 :
1300 OU$="[RVSON][YELLOW]OUT[RVSOF
F][GREEN]"
1310 :
1320 P$(3)="[YELLOW][HOME][DOWN][D
OWN][DOWN][DOWN][DOWN][DOWN][DOWN]
[DOWN][DOWN]
[DOWN][DOWN]"
1330 :
```



```
1340 P$(6)="[HOME][DOWN][DOWN][DOW
N][DOWN][DOWN][DOWN][DOWN][DOWN][D
OWN][DOWN][D
OWN][DOWN]"
1350 :
1360 P$(4)="[YELLOW][HOME][DOWN][D
OWN][DOWN][DOWN][DOWN][DOWN][DOWN]
[DOWN][DOWN]
[DOWN][DOWN]
                 "
1370 :
1380 P$(1)="[HOME][DOWN][DOWN][DOW
N][DOWN][DOWN][DOWN][DOWN][DOWN][D
OWN][DOWN][D
OWN][DOWN][DOWN][DOWN][DOWN][DOWN]
[DOWN][DOWN][DOWN][DOWN][DOWN][DOW
N][DOWN]"
1390 :
1400 P$(2)="[HOME][DOWN][DOWN][DOW
N][DOWN][DOWN][DOWN][DOWN][DOWN][D
OWN][DOWN][D
OWN][DOWN][DOWN][DOWN][DOWN][DOWN]
[DOWN][DOWN][DOWN][DOWN][DOWN][DOW
N][DOWN][DOW
N][DOWN]"
1410 DIMS$(28),C$(28),M$(28),M(28)
,R(28)
1420 DIMD$(9)
1430 :
1440 D$(1)="[c 5][c A][s C][s C][c
 S]"+U$
1450 :
1460 D$(2)="[s -]  [s -]"+U$
1470 D$(3)="[s -]  [s -]"+U$
1480 D$(4)="[s -]  [s -]"+U$
1490 D$(5)="[c Q][s C][s C][c W]"+
U$
1500 D$(6)="[s -]  [s -]"+U$
1510 D$(7)="[s -]  [s -]"+U$
1520 D$(8)="[s -]  [s -]"+U$
1530 :
1540 D$(9)="[c Z][s C][s C][c X]"+
U$
1550 :
1560 EM$(1)="     "+U$
1570 EM$(2)="     "+U$
1580 EM$(3)="     "+U$
1590 EM$(4)="     "+U$
1600 EM$(5)="     "+U$
1610 EM$(6)="     "+U$
1620 EM$(7)="     "+U$
1630 EM$(8)="     "+U$
1640 EM$(9)="     "+U$
1650 EM$(10)="     "+U$
1660 :
1670 BL$(1)="[c 5][c A][s C][s C][
c S]"+U$
1680 :
1690 BL$(2)="[s -][RVSON][s V][s V
][RVSOFF][s -]"+U$
1700 :
1710 BL$(3)="[s -][RVSON][s V][s V
][RVSOFF][s -]"+U$
1720 :
1730 BL$(4)="[s -][RVSON][s V][s V
][RVSOFF][s -]"+U$
1740 :
1750 BL$(5)="[s -][RVSON][s V][s V
][RVSOFF][s -]"+U$
1760 :
1770 BL$(6)="[s -][RVSON][s V][s V
][RVSOFF][s -]"+U$
1780 :
1790 BL$(7)="[s -][RVSON][s V][s V
][RVSOFF][s -]"+U$
1800 :
1810 BL$(8)="[c Z][s C][s C][c X]"
+U$
1820 DR$(1)="[CYAN][c A][s C][s C]
[s C][c R][s C][s C][s C][c S]"+U$
1830 :
1840 DR$(2)="[s -]   [s -]   [s -]
"+U$
1850 DR$(3)="[s -]   [s -]   [s -]
"+U$
1860 DR$(4)="[s -]   [s -]   [s -]
"+U$
1870 :
1880 DR$(5)="[c Z][s C][s C][s C][
c E][s C][s C][s C][c X]"+U$
1890 :
1900 MO$(1)="[BLUE]ME"
1910 :
1920 MO$(2)="[GREEN]YOU"
1930 NG=0
1940 S$(1)="0-0":S$(2)="0-1":S$(3)
="0-2":S$(4)="0-3":S$(5)="0-4":S$(
6)="0-5"
1950 S$(7)="0-6"
1960 S$(8)="1-1":S$(9)="1-2":S$(10
)="1-3":S$(11)="1-4":S$(12)="1-5":
S$(13)="1-6"
1970 S$(14)="2-2":S$(15)="2-3":S$(
16)="2-4":S$(17)="2-5":S$(18)="2-6
"
1980 S$(19)="3-3":S$(20)="3-4":S$(
21)="3-5":S$(22)="3-6"
1990 S$(23)="4-4":S$(24)="4-5":S$(
25)="4-6"
2000 S$(26)="5-5":S$(27)="5-6"
2010 S$(28)="6-6"
2020 :
2030 PRINT"[CLEAR]":T=0
2040 PRINT"[HOME][DOWN][DOWN][DOWN
][DOWN][DOWN][DOWN][DOWN][DOWN][DO
WN]":FORA=0T
03:PRINTTAB(9)DO$(A);:NEXTA
2050 FORB=1TO9
2060 :
```

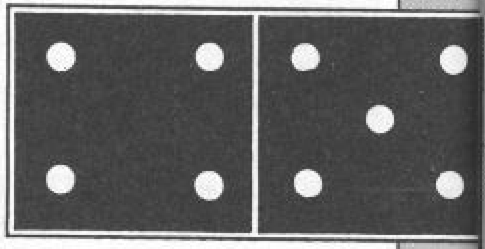


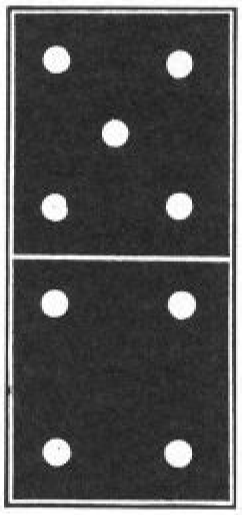
```
2070 PRINT"[HOME][UP]"
2080 FORA=1TO8:PRINTTAB(T)BL$(A);:
NEXTA:T=T+4
2090 GOSUB5180
2100 NEXTB
2110 T=0
2120 FORB=1TO9
2130 :
2140 PRINT"[HOME][DOWN][DOWN][DOWN
][DOWN][DOWN][DOWN][DOWN][DOWN][DO
WN][DOWN][DO
WN][DOWN][DOWN]"
2150 FORA=1TO9:PRINTTAB(T)D$(A);:N
EXTA:T=T+4
2160 GOSUB5180
2170 NEXTB
2180 FORM=1TO9
2190 R=INT(28*RND(0)+1)
2200 IFS$(R)=""THEN2190
2210 C$(M)=S$(R):M$(M)=S$(R)
2220 S$(R)=""
2230 NEXTM
2240 FORM=1TO9
2250 V1=VAL(LEFT$(C$(M),1))
2260 V2=VAL(RIGHT$(C$(M),1))
2270 IFM=1THENT=1:TT=1
2280 IFM=2THENT=5:TT=5
2290 IFM=3THENT=9:TT=9
2300 IFM=4THENT=13:TT=13
2310 IFM=5THENT=17:TT=17
2320 IFM=6THENT=21:TT=21
2330 IFM=7THENT=25:TT=25
2340 IFM=8THENT=29:TT=29
2350 IFM=9THENT=33:TT=33
2360 :
2370 PRINT"[HOME][DOWN][DOWN][DOWN
][DOWN][DOWN][DOWN][DOWN][DOWN][DO
WN][DOWN][DO
WN][DOWN][DOWN][DOWN]"
2380 IFV1<1THENV1=0:GOTO2440
2390 FORA=1TOV1
2400 JP=V1:GOSUB5510
2410 PRINTTAB(T)SP$;:T=T+1:IFT>TT+
1THENPRINT:T=TT
2420 NEXTA
2430 :
2440 PRINT"[HOME][DOWN][DOWN][DOWN
][DOWN][DOWN][DOWN][DOWN][DOWN][DO
WN][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN][DOWN]":T=TT
2450 IFV2<1THENV2=0:GOTO 2500
2460 FORZ=1TOV2
2470 JP=V2: GOSUB5510
2480 PRINTTAB(T)SP$;:T=T+1:IFT>TT+
1THENPRINT :T=TT
2490 NEXTZ
2500 NEXTM
2510 :
```



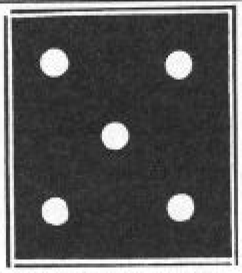
```
2520 FORX=1TO9
2530 R=INT(28*RND(0)+1)
2540 IFS$(R)=""THEN2530
2550 X$(X)=S$(R)
2560 S$(R)=""
2570 NEXTX
2580 :
2590 :
2600 :
2610 PRINTP$(1)TAB(1)"1"TAB(6)"2"T
AB(10)"3"TAB(14)"4"TAB(18)"5"TAB(2
2)"6"
2620 PRINTP$(1)TAB(25)"7"TAB(29)"8
"TAB(34)"9"
2630 GOTO4080:
2640 :
2650 :
2660 PRINT"[YELLOW][HOME][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN][DOWN][
DOWN]"TAB(0)"ENTER":PRINTTAB(0)"YO
UR":PRINTTAB(0)"MOVE"
2670 POKE198,0
2680 GETG$:IFG$=""THEN2680
2690 IFG$=CHR$(13)THENGOSUB5410
2700 IFG$=CHR$(13)THENGOSUB5630:IF
CH=1THENCH=0:GOTO2680
2710 IFG$=CHR$(13)THENZZ=1:MO$(2)=
MO$(2)+"[DOWN][LEFT][LEFT][LEFT]"+
OU$:PRINTP$(
6)TAB(36)MO$(2):GOTO3140
2720 G=VAL(G$):IFG<1ORG>9THEN2680
2730 :
2740 PRINT"[YELLOW][HOME][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN][DOWN][
DOWN]"TAB(0)"     ":PRINTTAB(0)"
    ":PRINTTAB(0)"     "
2750 IFNG=0THEN PRINTTAB(10)P$(4)
2760 IFM$(G)=""THEN2660
2770 V1=VAL(LEFT$(M$(G),1))
2780 V2=VAL(RIGHT$(M$(G),1))
2790 IFNG=1ANDV1<>LDANDV1<>RDANDV2
<>LDANDV2<>RDTHEN2660
2800 :
2810 MO$(2)=MO$(2)+"[DOWN][LEFT][L
EFT][LEFT]"+M$(G)
2820 PRINTP$(6)TAB(36)MO$(2)
2830 :
2840 PRINT"[YELLOW][HOME][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN]"
2850 FORA=1TO5:PRINTTAB(15)DR$(A);
:NEXTA
2860 IFNG=0THEN2940
2870 GOTO4880
2880 :
2890 K=10:EE$="[HOME][DOWN][DOWN][
DOWN][DOWN][DOWN][DOWN][DOWN][DOWN
```

```
][DOWN][DOWN
][DOWN][DOWN][DOWN][DOWN]":GOSUB37
00:GOSUB5180:M$(G)=""
2900 IFV1=LDTHENLD=V2:GOSUB5110:GO
SUB3510:GOSUB3830:GOTO3140
2910 IFV2=LDTHENLD=V1:GOSUB5110:GO
SUB3510:GOSUB3830:GOTO3140
2920 IFV2=RDTHENRD=V1:GOSUB5110:GO
SUB3510:GOSUB3830:GOTO3140
2930 IFV1=RDTHENRD=V2:GOSUB5110:GO
SUB3510:GOSUB3830:GOTO3140
2940 :
2950 :
2960 PRINT"[YELLOW][HOME][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN][DOWN]"
:T=16:GOSUB5180
2970 IFV1<1THEN3020
2980 FORA=1TOV1
2990 JP=V1:GOSUB5510
3000 PRINTTAB(T)SP$;:T=T+2:IFT>18T
HENPRINT:T=16
3010 NEXTA
3020 T=20
3030 :
3040 PRINT"[YELLOW][HOME][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN][DOWN]"
3050 IFV2<1THEN3100
3060 FORZ=1TOV2
3070 JP=V2:GOSUB5510
3080 PRINTTAB(T)SP$;:T=T+2:IFT>22T
HENPRINT:T=20
3090 NEXTZ
3100 LD=V1:RD=V2:NG=1:GOSUB5180:GO
SUB5110
3110 :
3120 K=10:EE$="[HOME][DOWN][DOWN][
DOWN][DOWN][DOWN][DOWN][DOWN][DOWN
][DOWN][DOWN
][DOWN][DOWN][DOWN][DOWN]":GOSUB37
00:M$(G)="":GOSUB5180:GOTO3140
3130 :
3140 :
3150 :
3160 GOTO5820:
3170 FORA=1TO9
3180 IFX$(A)=""THEN3250
3190 C1(A)=VAL(LEFT$(X$(A),1))
3200 C2(A)=VAL(RIGHT$(X$(A),1))
3210 IFC1(A)=LDTHENLD=C2(A):GOTO33
30
3220 IFC1(A)=RDTHENRD=C2(A):GOTO33
30
3230 IFC2(A)=LDTHENLD=C1(A):GOTO33
30
3240 IFC2(A)=RDTHENRD=C1(A):GOTO33
30
3250 NEXTA
3260 :
3270 MO$(1)=MO$(1)+"[DOWN][LEFT][L
EFT][LEFT]"+"[RVSON][YELLOW]OUT[RV
SOFF][BLUE]"
3280 :
3290 PRINT"[HOME][DOWN]"TAB(37)MO$
(1):GOSUB5410
3300 FORDELAY=1TO200:NEXTDELAY
3310 IFZZ=1THENGOTO4560
3320 GOTO2660
3330 GOSUB5110
3340 :
3350 MO$(1)=MO$(1)+"[DOWN][LEFT][L
EFT][LEFT]"+X$(A)
3360 :
3370 PRINT"[HOME][DOWN]"TAB(37)MO$
(1)
3380 IFA=1THENT=0
3390 IFA=2THENT=4
3400 IFA=3THENT=8
3410 IFA=4THENT=12
3420 IFA=5THENT=16
3430 IFA=6THENT=20
3440 IFA=7THENT=24
3450 IFA=8THENT=28
3460 IFA=9THENT=32
3470 X$(A)=""
3480 :
3490 ZZ=0:K=9:EE$="[HOME]":GOSUB37
90:GOSUB5180:GOSUB3510:GOSUB3960:G
OTO2640
3500 :
3510 PRINT"[YELLOW][HOME][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN]"
3520 FORA=1TO5:PRINTTAB(15)DR$(A);
:NEXTA
3530 :
3540 PRINT"[YELLOW][HOME][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN][DOWN]"
:T=16
3550 IFLD<1THEN3610
3560 FORA=1TOLD
3570 JP=LD:GOSUB5510
3580 PRINTTAB(T)SP$;:T=T+2:IFT>18T
HENPRINT:T=16
3590 NEXTA
3600 :
3610 PRINT"[YELLOW][HOME][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN][DOWN]"
:T=20
3620 IFRD<1THEN3680
3630 FORA=1TORD
3640 JP=RD:GOSUB5510
3650 PRINTTAB(T)SP$;:T=T+2:IFT>22T
HENPRINT:T=20
3660 NEXTA
```



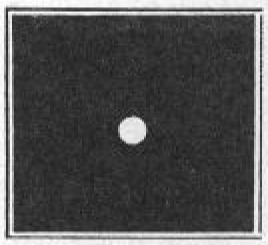


```
3670 GOSUB5180
3680 RETURN
3690 :
3700 IFG=1THENT=0
3710 IFG=2THENT=4
3720 IFG=3THENT=8
3730 IFG=4THENT=12
3740 IFG=5THENT=16
3750 IFG=6THENT=20
3760 IFG=7THENT=24
3770 IFG=8THENT=28
3780 IFG=9THENT=32
3790 PRINTEE$;
3800 FORA=1TOK
3810 PRINTTAB(T)EM$(A);:NEXTA
3820 RETURN
3830 FORU=1TO9
3840 IFM$(U)=""THENTU=TU+1:IFTU=9T
HEN3880
3850 NEXTU
3860 TU=0:RETURN
3870 :
3880 PRINT"[HOME][DOWN][DOWN][DOWN
][DOWN][DOWN][DOWN][DOWN][DOWN][DO
WN][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN][DOWN][DOWN]                WELL
 PLAYED":WN=
0:LF=17:TP=1:GOSUB5330
3890 :
3900 :
3910 PRINT:PRINT:PRINTTAB(10)"ANOT
HER GAME [Y-N]"
3920 GETA$:IFA$=""THEN3920
3930 IFA$="Y"THENRUN1210
3940 IFA$="N"THENEND
3950 GOTO3920
3960 FORU=1TO9
3970 IFX$(U)=""THENTU=TU+1:IFTU=9T
HEN4010
3980 NEXTU
3990 TU=0:RETURN
4000 :
4010 PRINT"[HOME][DOWN][DOWN][DOWN
][CYAN]                 HARD LUCK":WN
=17:LF=0:TP=
-1:GOSUB5330
4020 :
4030 :
4040 :
4050 PRINT:PRINT:PRINTTAB(10)"ANOT
HER GAME [Y-N]"
4060 GOTO3920
4070 :
4080 ND=INT(RND(.5)*2+1)
4090 PRINT"[HOME][DOWN][DOWN][DOWN
][DOWN][DOWN][DOWN][DOWN][DOWN][DO
WN]":FORA=1T
O4:PRINTTAB(9)K1$:NEXTA
4100 IFND=1THENND=0:PRINTP$(3)TAB(
10)"YOU HAVE FIRST DROP":GOTO2640
4110 PRINTP$(3)TAB(8)"I HAVE THE F
IRST DROP":FORDELAY=1TO1000:NEXTDE
LAY
4120 PRINTTAB(8)P$(4)
4130 FORND=1TO9
4140 IFLEFT$(X$(ND),1)=RIGHT$(X$(N
D),1)THEN4180
4150 NEXTND
4160 ND=INT(RND(.5)*6+1)
4170 :
4180 PRINT"[YELLOW][HOME][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN]"
4190 FORA=1TO5:PRINTTAB(15)DR$(A);
:NEXTA
4200 :
4210 K=9:EE$="[HOME]":GOSUB4450:GO
SUB5180:GOSUB3790
4220 :
4230 MO$(1)=MO$(1)+"[DOWN][LEFT][L
EFT][LEFT]"+X$(ND)
4240 :
4250 PRINT"[HOME][DOWN]"TAB(37)MO$
(1)
4260 C1=VAL(LEFT$(X$(ND),1))
4270 C2=VAL(RIGHT$(X$(ND),1))
4280 :
4290 PRINT"[YELLOW][HOME][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN][DOWN]"
:T=16
4300 IFC1<1THEN4350
4310 FORA=1TOC1
4320 JP=C1:GOSUB5510
4330 PRINTTAB(T)SP$;:T=T+2:IFT>18T
HENPRINT:T=16
4340 NEXTA
4350 T=20
4360 :
4370 PRINT"[YELLOW][HOME][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN][DOWN]"
4380 IFC2<1THEN4430
4390 FORZ=1TOC2
4400 JP=C2:GOSUB5510
4410 PRINTTAB(T)SP$;:T=T+2:IFT>22T
HENPRINT:T=20
4420 NEXTZ
4430 X$(ND)="":GOSUB5180:LD=C1:RD=
C2:ND=0:GOSUB5110
4440 IFNG=0THENNG=1:GOTO2640
4450 :
4460 IFND=1THENT=0
4470 IFND=2THENT=4
4480 IFND=3THENT=8
4490 IFND=4THENT=12
4500 IFND=5THENT=16
```

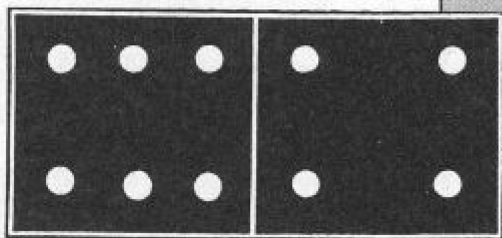


```
4510 IFND=6THENT=20
4520 IFND=7THENT=24
4530 IFND=8THENT=28
4540 IFND=9THENT=32
4550 RETURN
4560 C1=0:C2=0:V1=0:V2=0
4570 :
4580 PRINT"[CLEAR]":FORA=0TO3:PRIN
TTAB(10)DO$(A);:NEXTA
4590 :
4600 PRINT"[DOWN]"TAB(12)"[WHITE]S
TITCHED  GAME"
4610 :
4620 PRINT"[HOME][DOWN][DOWN][DOWN
][DOWN][DOWN][DOWN][DOWN][DOWN][DO
WN][DOWN][DO
WN][DOWN][DOWN][DOWN][YELLOW]"TAB(
36)"MINE"
4630 FORA=1TO9
4640 IFX$(A)=""THEN4680
4650 C1=C1+VAL(LEFT$(X$(A),1))
4660 C2=C2+VAL(RIGHT$(X$(A),1))
4670 PRINTTAB(36)X$(A)
4680 NEXTA
4690 :
4700 PRINT"[HOME][DOWN][DOWN][DOWN
][DOWN][DOWN][DOWN][DOWN][DOWN][DO
WN][DOWN][DO
WN][DOWN][DOWN][DOWN][YELLOW]YOURS
[c 7]"
4710 FORA=1TO9
4720 IFM$(A)=""THEN4760
4730 V1=V1+VAL(LEFT$(M$(A),1))
4740 V2=V2+VAL(RIGHT$(M$(A),1))
4750 PRINTTAB(0)M$(A)
4760 NEXTA
4770 :
4780 PRINT"[YELLOW][HOME][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN][DOWN][
DOWN]"TAB(26)"I ":PRINTTAB(25)"HAV
E":PRINTTAB(25)C1+C2" "
4790 :
4800 PRINT"[YELLOW][HOME][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN][DOWN][
DOWN]"TAB(10)"YOU ":PRINTTAB(10)"H
AVE":PRINTTAB(10)V1+V2" "
4810 :
4820 IFC1+C2<V1+V2THENPRINT:PRINT:
PRINTTAB(16)"I WIN":GOTO4030
4830 :
4840 IFV1+V2<C1+C2THENPRINT:PRINT:
PRINTTAB(16)"YOU WIN":GOTO3890
4850 :
4860 IFV1+V2=C1+C2THENPRINT:PRINT:
PRINTTAB(16)"WE DREW":GOTO3890
4870 :
4880 IFV1=LDANDV2=RDTHEN4910
4890 IFV1=RDANDV2=LDTHEN4910
4900 GOTO2890
4910 IFLD=RDTHEN2890
4920 :
4930 GOSUB5740:GOSUB5740:GOSUB5740
4940 PRINT"[YELLOW][HOME][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN][DOWN]"
TAB(0)GU$(LD):PRINTTAB(0)"UP OR "
4950 PRINTTAB(0)GU$(RD):PRINTTAB(0
)"UP   "
4960 GOSUB5740:GOSUB5740:GOSUB5740
4970 GETSD$:IFSD$=""THEN4970
4980 SD=VAL(SD$)
4990 IFSD<>LDANDSD<>RDTHEN4970
5000 GOSUB5740:GOSUB5740:GOSUB5740
5010 :
5020 PRINT"[YELLOW][HOME][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN][DOWN]"
TAB(0)"        ":PRINTTAB(0)"        "
:PRINTTAB(0)"        "
5030 PRINTTAB(0)"        ":PRINTTAB
(0)"        "
5040 :
5050 K=10:EE$="[HOME][DOWN][DOWN][
DOWN][DOWN][DOWN][DOWN][DOWN][DOWN
][DOWN][DOWN
][DOWN][DOWN][DOWN][DOWN]"
5060 IFSD=V1THENRD=V1:LD=V1:GOSUB5
110:GOSUB3510:GOSUB3830:GOSUB3700:
M$(G)="":GOS
UB3830
5070 IFSD=V1THENGOTO3140
5080 IFSD=V2THENLD=V2:RD=V2:GOSUB5
110:GOSUB3510:GOSUB3830:GOSUB3700:
M$(G)="":GOS
UB3830
5090 IFSD=V2THENGOTO3140
5100 GOTO4910
5110 VL=LD:VR=RD
5120 :
5130 PRINT"[YELLOW][HOME][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN][DOWN][
DOWN][DOWN]"TAB(9)GU$(VL)
5140 :
5150 PRINT"[YELLOW][HOME][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN][DOWN][
DOWN][DOWN]"TAB(26)GU$(VR)
5160 RETURN
5170 :
5180 :
5190 SS=54272:LF=SS:HF=SS+1:AD=SS+
5
5200 SR=SS+6:WF=SS+4:VO=SS+24
5210 H3=SS+15:L3=SS+14
5220 FORX=SSTOSS+24
```

```
5230 POKEX,0
5240 NEXTX
5250 POKEVO,15:POKEAD,5 :POKESR,5
5260 POKEWF,17
5270 FORTZ=1TO4
5280 POKEWF,130 :POKEHF,33:POKELF,
33
5290 POKEHF,0:POKELF,0
5300 NEXTTZ
5310 RETURN
5320 :
5330 POKE54296,15
5340 POKE54276,0:POKE54272,0:POKES
4273,0:POKE54277,0:POKE54278,240
5350 POKE54276,33
5360 FORZ=WNTOLFSTEPTP:POKE53280,Z
:FORZ1=1TO15:POKE54273,Z*Z1:NEXTZ1
,Z
5370 POKE54276,0
5380 POKE53280,0
5390 RETURN
5400 :
5410 POKE54296,15
5420 POKE54276,0:POKE54272,0:POKES
4273,0:POKE54277,0:POKE54278,240
5430 POKE54276,17
5440 FORZ=60TO0STEP-1
5450 POKE54273,Z:NEXTZ
5460 POKE54276,0
5470 POKE53280,0
5480 RETURN
5490 :
5500 :
5510 IFJP=1THENSP$="[RED][s Q][BLU
E]":RETURN
5520 :
5530 IFJP=2THENSP$="[YELLOW][s Q][
BLUE]":RETURN
5540 :
5550 IFJP=3THENSP$="[GREEN][s Q][B
LUE]":RETURN
5560 :
5570 IFJP=4THENSP$="[CYAN][s Q][BL
UE]":RETURN
5580 :
5590 IFJP=5THENSP$="[PURPLE][s Q][
BLUE]":RETURN
5600 :
5610 IFJP=6THENSP$="[c 1][s Q][BLU
E]":RETURN
5620 :
5630 FORU=1TO9:IFM$(U)=""THEN5670
5640 O1=VAL(LEFT$(M$(U),1))
5650 O2=VAL(RIGHT$(M$(U),1))
5660 IFO1=LDORO2=LDORO1=RDORO2=RDT
HEN5700
5670 NEXTU
5680 CH=0:RETURN
5690 :
```



```
5700 PRINT"[YELLOW][HOME][DOWN][DO
WN][DOWN][DOWN][DOWN][DOWN][DOWN][
DOWN][DOWN][
DOWN]"TAB(0)"[RVSON]YOU[RVSOFF]   "
:PRINTTAB(0)"[RVSON]CAN[RVSOFF]   "
5710 :
5720 PRINTTAB(0)"[RVSON]GO [RVSOFF
] ":PRINTTAB(0)"     "
5730 CH=1:GOSUB5410:GOSUB5410:RETU
RN
5740 POKE54296,15
5750 POKE54276,0:POKE54272,0:POKES
4273,0:POKE54277,0:POKE54278,240
5760 POKE54276,17
5770 NR=INT((250-0+1)*RND(.5)+0)
5780 POKE54273,NR
5790 POKE54276,0
5800 POKE53280,0
5810 RETURN
5820 FORA=1TO9
5830 IFX$(A)=""THEN5900
5840 C1(A)=VAL(LEFT$(X$(A),1))
5850 C2(A)=VAL(RIGHT$(X$(A),1))
5860 IFC1(A)=C2(A)ANDC2(A)=LDTHENL
D=C2(A):GOTO3330
5870 IFC1(A)=C2(A)ANDC2(A)=RDTHENR
D=C2(A):GOTO3330
5880 IFC2(A)=C1(A)ANDC1(A)=LDTHENL
D=C1(A):GOTO3330
5890 IFC2(A)=C1(A)ANDC1(A)=RDTHENR
D=C1(A):GOTO3330
5900 NEXTA
5910 GOTO3170
5920 :
5930 U$=CHR$(13)
5940 K1$="
   "
5950 :
5960 DO$(0)="[WHITE][c A][s C][s I
][s U][s C][s I][s U][c S][c A][s
I][c R][s U]
[s C][s I][s U][s C][s I][c A][s C
][c A][s C][c S]"+U$
5970 DO$(1)="[YELLOW][s -] [s -][s
 -] [s -][s -][s -][s -][s -][s -]
[s -] [s -][
s -] [s -][c Q][s C][c Z][s C][c S
]"+U$
5980 DO$(2)="[BLUE][c Z][s C][s K]
[s J][s C][s K][c Z]  [c X][c E][c
 Z] [c X][s
J][s C][s K][c Z][s C]  [WHITE][s
-]"+U$
5990 DO$(3)="[s C][s C][s C][s C][
s C][s C][s C][s C][s C][s C][s C]
[s C][s C][s
 C][s C][s C][s C][s C][s C][s C][
s C][c X][CYAN]"+U$
6000 RETURN
```



INPUT

Our resident expert answers your programming queries.

INPUT

Most computers with function keys enable you to redefine the keys in Basic. I have been trying to do this for some time on my C64 but without too much success. Could you possibly tell me how this could be done?
Norman Doyle
Cleveland

OUTPUT

Unfortunately, as you have no doubt found out, there is no simple way to define the C64. Most Basic extension packages (see our August and September issues) have a command which allows you to program keys by means of a new keyword. It is however quite simple to use these keys in your own programs without having to go to any expense. Take a look at the following short program

```
10 GET A$
20 IF A$ < CHR$(133) THEN 10
30 IF A$ > CHR$(140) THEN 10
40 A=(ASC(A$)-132*2-1:IF A > 8
THEN A=A-7
50 PRINT "FUNCTION KEY
";A;" IS PRESSED"
60 GOTO 10
```

This short routine will scan for all eight function keys. By adding the following line it is also possible to add four function keys by using the Commodore key as an extra shift key.

```
45 IF PEEK (653)=2 THEN
A=A/2+8
```

Reading the function keys in this manner is extremely simple as each of the keys returns a unique number to the GET instruction in line 10.

Below is a program which will allow you to actually store strings of characters on each key, the strings can either be keywords or strings of your own choosing.

Sixteen re-definable keys are catered for and these are obtained by pressing F1,F3,F5 and F7 with no shift, the Commodore key or the control key.

The whole program is based on interrupts and can be switched on by:

SYS 49152

and off by:

SYS 49155

When the program is first loaded the keys are set to:

F1 SYS 49158
F2 SYS 49155
F3 RUN + RETURN
F4 POKE
F5 LIST
F6 PEEK(
F7 LIST + RETURN
F8 ?PEEK(
F9 MID$(
F10 LEFT$(
F11 RIGHT$(
F12 LEN(
F13 53248
F14 54272
F15 LOAD
F16 SAVE

The function of each key can be changed by either pressing F1 or typing SYS 49158 followed by the text in quotation marks, a comma and then the key number. The text stored on each key can be no more than 10 characters and the key number no greater than 16. F1 and F2 cannot be changed since they hold the important SYS calls.

Here are a few examples:

SYS 49158 "P[s 0]53281,0",7

This will POKE 53281 with 0.

SYS 49158 "LOAD"+CHR$(34)+"S"+CHR$(34)+"8 ←",5

This will type LOAD "$",8 and execute a return.

If you wish to add a RETURN onto the end of any function you must put a left hand arrow as the last character in the text when defining the function.

Below you will find the M/C source file and a basic loader.

Machine code

```
1000 ;******************************
1010 ;***                        ***
1020 ;***  DEFINE FUNCTION KEYS  ***
1030 ;***                        ***
1040 ;******************************
1050 ;
1060 ;
1070 *       = 49152
1080 IRQOUT  = 59953
1090 KEYSKN  = 197
1100 KEYBUF  = 631
1110 KEYQUE  = 198
1120 IRQVEC  = 788
1130 PRINT   = 65490
1140 SHIFT   = 653
1150 BASIC1  = 44797
1160 BASIC2  = 44426
1170 BASIC3  = 47095
1180 BASIC4  = 44446
1190 BASIC5  = 46755
1200 NUMBER  = 20
1210 STRING  = 34
1220         ;
1230         ;
1240 JMPTB2  JMP IRQON
1250 JMPTB3  JMP IRQOFF
1260 JMPTB1  JMP KEYDEF
1270         ;
1280         ;
1290 IRQON   SEI            ;SWITCH ON NEW
1300         LDA #<IRQ      ;IRQ (F KEYS)
1310         STA IRQVEC
1320         LDA #>IRQ
1330         STA IRQVEC+1
1340         CLI
1350         RTS
1360         ;
1370         ;
1380 IRQOFF  SEI            ;SWITCH OFF
1390         LDA #<IRQOUT   ;NEW IRQ AND
1400         STA IRQVEC     ;REPLACE THE
1410         LDA #>IRQOUT   ;OLD VECTORS
1420         STA IRQVEC+1
1430         CLI
1440         RTS
1450         ;
1460         ;
1470 IRQ     LDA KEYSKN     ;CHECK TO SEE
1480         CMP TEMP1      ;IF KEY HAS
```

Machine code

```
1490          BNE OK         ;REPEATED
1500          JMP IRQOUT
1510 OK       LDA SHIFT      ;CHECK FOR TWO
1520          BEQ NOERR      ;OR MORE SHIFT
1530          CMP #1         ;KEYS PRESSED
1540          BEQ NOERR
1550          CMP #2
1560          BEQ NOERR
1570          CMP #4
1580          BEQ NOERR1
1590          JMP IRQOUT     ;FOUND MORE
1600                         ;THAN ONE KEY
1610                         ;IS PRESSED
1620          ;
1630 NOERR1   LDA #3         ;STORE ALL THE
1640 NOERR    STA TEMP3      ;INFO
1650          LDA KEYSKN
1660          STA TEMP1
1670          CMP #3
1680          BNE NOTODD
1690          LDA #7
1700 NOTODD   SEC
1710          SBC #4
1720          STA TEMP2
1730          LDA KEYSKN
1740          CMP #3
1750          BCS OK03
1760          JMP IRQOUT
1770 OK03     CMP #7
1780          BCC OK04
1790          JMP IRQOUT
1800          ;
1810 OK04     LDA #0         ;FIND THE TEXT
1820          STA TEMP4      ;FOR EACH KEY
1830          LDX TEMP2      ;PRESSED
1840 NXTKEY   BEQ FOUNDK
1850          LDA TEMP4      ;ADD 10 TO THE
1860          CLC            ;POINTER
1870          ADC #10        ;EG. 10 CHARS
1880          STA TEMP4      ;PER KEY
1890          DEX
1900          JMP NXTKEY
1910 FOUNDK   LDX TEMP3
1920 NXTSHF   BEQ FOUNDS
1930          LDA TEMP4      ;ADD 40 TO THE
1940          CLC            ;POINTER
1950          ADC #40        ;EG. 40 CHARS
1960          STA TEMP4      ;PER FOUR KEYS
1970          DEX            ;EQU 1 SHIFT
1980          JMP NXTSHF
1990          ;
2000          ;TEXT NOW FOUND
2010          ;
2020          ;FILL BUFFER WITH TEXT
2030          ;
2040 FOUNDS   LDX #0         ;PUT DATA INTO
2050          LDY TEMP4      ;KEYBOARD
2060 NXTCHR   LDA DATA,Y     ;BUFFER....
2070          CMP #'←'       ;FIND '←' AND
2080          BNE NORETN     ;REPLACE WITH
2090          LDA #13        ;(RETURN)
2100 NORETN   STA KEYBUF,X
2110          INY
2120          INX
2130          CPX #10
2140          BCC NXTCHR
2150          STX KEYQUE     ;SET QUEUE LEN
2160          JMP IRQOUT     ;ALL DONE!
2170 KEYDEF   JSR BASIC4     ;PULL STRING
2180          JSR BASIC5     ;FROM INPUT
2190          CMP #11        ;CHECK LEN
2200          BCC LENOK
2210 SYNTAX   LDX #0         ;PRINT ERROR
2220 NEXTCR   LDA MESSAG,X
2230          BEQ DONE
2240          JSR PRINT
2250          INX
2260          JMP NEXTCR
2270 DONE     RTS
2280 LENOK    STA TEMP5      ;STORE LEN
2290          LDX #0         ;RESET TMPKEY
2300          LDA #0
2310 BACK01   STA TMPKEY,X
2320          INX
2330          CPX #10
2340          BCC BACK01
2350          LDY #0         ;STORE STRING
2360 BACK02   LDA (STRING),Y
2370          STA TMPKEY,Y
2380          INY
2390          CPY TEMP5
2400          BCC BACK02
2410          JSR BASIC1     ;GET NUMBER
2420          JSR BASIC2     ;INPUT
2430          JSR BASIC3     ;EG. KEY NO.
2440          LDA NUMBER
2450          CMP #3
2460          BCC ERROR
2470          SEC
2480          SBC #1
2490          CMP #16        ;CHECK NO.
2500          BCC RITNUM
2510 ERROR    JMP SYNTAX
2520 RITNUM   CMP #7         ;FIND WHERE
2530          BCS NONEED     ;TO PUT
2540          TAX            ;INFO FOR
2550          LDA DUMMY,X    ;KEY
2560 NONEED   TAX
2570          LDA #0
2580          STA TEMP5
```

INPUT

Machine code (continued)

```
2590 WOW     CPX #0
2600         BEQ FOUND
2610         LDA TEMP5       ;ADD TEN FOR
2620         CLC             ;EACH KEY
2630         ADC #10
2640         STA TEMP5
2650         DEX
2660         JMP WOW
2670 FOUND   LDX #0          ;STORE IN ITS
2680         LDY TEMP5       ;RIGHT PLACE
2690 LAST    LDA TMPKEY,X
2700         STA DATA,Y
2710         INX
2720         INY
2730         CPX #10
2740         BCC LAST
2750         RTS
2760         ;
2770         ;
2780 TEMP1   BRK
2790 TEMP2   BRK
2800 TEMP3   BRK
2810 TEMP4   BRK
2820 TEMP5   BRK
2830         ;
2840         ;
2850 DATA    ;FOR THE KEYS
2860 .BYT 'SYS49158 ',0          ;KEY1
2870 .BYT 'RUN←',0,0,0,0,0,0    ;KEY3
2880 .BYT 'LIST',0,0,0,0,0,0    ;KEY5
2890 .BYT 'LIST←',0,0,0,0,0     ;KEY7
2900     ;
2910     ;NOW F1-F7 WITH SHIFT
2920     ;
2930 .BYT 'SYS49155',0,0        ;KEY1
2940 .BYT '?PEEK(',0,0,0,0      ;KEY3
2950 .BYT 'GOTO',0,0,0,0,0,0    ;KEY5
2960 .BYT 'GOSUB',0,0,0,0,0     ;KEY7
2970     ;
2980     ;NOW F1-F7 WITH CBM KEY 'C='
2990     ;
3000 .BYT 'MID$(',0,0,0,0,0     ;KEY1
3010 .BYT 'LEFT$(',0,0,0,0      ;KEY3
3020 .BYT 'RIGHT$(',0,0,0       ;KEY5
3030 .BYT 'LEN(',0,0,0,0,0,0    ;KEY7
3040     ;
3050     ;NOW F1-F7 WITH CTRL KEY
3060     ;
3070 .BYT '53248',0,0,0,0,0     ;KEY1
3080 .BYT '54272',0,0,0,0,0     ;KEY3
3090 .BYT 'LOAD ',0,0,0,0,0     ;KEY5
3100 .BYT 'SAVE ',0,0,0,0,0     ;KEY7
3110     ;
3120 TMPKEY .BYT 0,0,0,0,0,0,0,0
3130 DUMMY  .BYT 0,4,1,5,2,6,3,7
3140 MESSAG .BYT 'ERROR IN INPUT',0
3150 .END
```

Basic Loader

```
100 POKE53280,0:POKE53281,0
110 PRINT"[CLEAR][RIGHT][RIGHT][RI
GHT][RIGHT][c 5]THIS IS THE BASIC
LOADER FOR"
120 PRINT"[DOWN]     16 FUNCTION K
EYS & DEFINER
130 PRINT"[DOWN]          (C) A.CROW
THER 1985
140 PRINT"[DOWN][DOWN] [WHITE]  TO
 BOOT      - SYS 49152
150 PRINT"[DOWN]TO SWITCH OFF   - S
YS 49155 OR F2
160 PRINT"[DOWN]  TO DEFINE     - S
YS 49158 "CHR$(34)"TEXT"CHR$(34)",
N[s W]
170 PRINT"[DOWN][c 4] "CHR$(34)"TE
XT"CHR$(34)" = 0 TO 10 CHARS LONG
180 PRINT"[DOWN] N[s W]       - 3 TO
 16 FUNTION NUMBER
190 PRINT"[DOWN][RED]          ^[c
5]NOTE 1 & 2 ARE UNCHANGEABLE
200 PRINT"[BLUE][DOWN][DOWN][DOWN]
 LOADING ........"
205 C=494:C$="[BLACK][c 4][c 5][c
8][WHITE][c 8][c 5][c 4]":D=1
210 LI=1000:FORI=49152TO49645STEP9
220 T=0:FORJ=0TO8:READA:T=T+A:POKE
I+J,A
225 PRINT"[HOME][DOWN][DOWN][DOWN]
[DOWN][DOWN][DOWN][DOWN][DOWN][DOW
N][DOWN][DOW
N][DOWN][DOWN][DOWN][DOWN][DOWN][D
OWN][DOWN][DOWN][DOWN][DOWN]"TAB(2
0)C"[LEFT] "
;:C=C-1
226 D=D+1:IFD=9THEND=1
227 PRINTMID$(C$,D,1):NEXT
230 READA:IFA<>TTHENPRINT"[CLEAR][
WHITE]ERROR IN ";LI:PRINT"[BLACK]L
IST"LI"[WHIT
E]":GOTO260
240 LI=LI+10:NEXT
250 POKE53281,1:POKE53280,1:PRINT"
[CLEAR][BLACK]";:END
260 POKE198,3:POKE631,19:POKE632,1
7:POKE633,13:END
```

Basic Loader continued

```
1000 DATA76,9,192,76,22,192,76,172
,192,1007
1010 DATA120,169,35,141,20,3,169,1
92,141,990
1020 DATA21,3,88,96,120,169,49,141
,20,707
1030 DATA3,169,234,141,21,3,88,96,
165,920
1040 DATA197,205,43,193,208,3,76,4
9,234,1208
1050 DATA173,141,2,240,17,201,1,24
0,13,1028
1060 DATA201,2,240,9,201,4,240,3,7
6,976
1070 DATA49,234,169,3,141,45,193,1
65,197,1196
1080 DATA141,43,193,201,3,208,2,16
9,7,967
1090 DATA56,233,4,141,44,193,165,1
97,201,1234
1100 DATA3,176,3,76,49,234,201,7,1
44,893
1110 DATA3,76,49,234,169,0,141,46,
193,911
1120 DATA174,44,193,240,13,173,46,
193,24,1100
1130 DATA105,10,141,46,193,202,76,
111,192,1076
1140 DATA174,45,193,240,13,173,46,
193,24,1101
1150 DATA105,40,141,46,193,202,76,
129,192,1124
1160 DATA162,0,172,46,193,185,48,1
93,201,1200
1170 DATA95,208,2,169,13,157,119,2
,200,965
1180 DATA232,224,10,144,238,134,19
8,76,49,1305
1190 DATA234,32,158,173,32,163,182
,201,11,1186
1200 DATA144,15,162,0,189,224,193,
240,7,1174
1210 DATA32,210,255,232,76,184,192
,96,141,1418
1220 DATA47,193,162,0,169,0,157,20
8,193,1129
1230 DATA232,224,10,144,248,160,0,
177,34,1229
1240 DATA153,208,193,200,204,47,19
3,144,245,1587
1250 DATA32,253,174,32,138,173,32,
247,183,1264
1260 DATA165,20,201,3,144,7,56,233
,1,830
1270 DATA201,16,144,3,76,182,192,2
01,7,1022
1280 DATA176,4,170,189,216,193,170
,169,0,1287
1290 DATA141,47,193,224,0,240,13,1
73,47,1078
1300 DATA193,24,105,10,141,47,193,
202,76,991
1310 DATA8,193,162,0,172,47,193,18
9,208,1172
1320 DATA193,153,48,193,232,200,22
4,10,144,1397
1330 DATA244,96,64,60,0,10,70,83,8
9,716
1340 DATA83,52,57,49,53,56,32,0,82
,464
1350 DATA85,78,95,0,0,0,0,0,258
1360 DATA76,73,83,84,0,0,0,0,316
1370 DATA0,76,73,83,84,95,0,0,0,41
1
1380 DATA0,0,83,89,83,52,57,49,53,
466
1390 DATA53,0,0,63,80,69,69,75,40,
449
1400 DATA0,0,0,71,79,84,79,0,313
1410 DATA0,0,0,0,70,82,69,68,289
1420 DATA0,0,0,0,0,77,73,68,218
1430 DATA36,40,0,0,0,0,0,76,69,221
1440 DATA70,84,36,40,0,0,0,0,82,31
2
1450 DATA73,71,72,84,36,40,0,0,0,3
76
1460 DATA76,69,78,40,0,0,0,0,0,263
1470 DATA0,53,51,50,52,56,0,0,0,26
2
1480 DATA0,0,53,52,50,55,50,0,0,26
0
1490 DATA0,0,0,76,79,65,68,32,0,32
0
1500 DATA0,0,0,0,83,65,86,69,32,33
5
1510 DATA0,0,0,0,0,70,82,69,68,289
1520 DATA0,0,0,0,0,0,1,5,2,8
1530 DATA6,3,7,69,82,82,79,82,32,4
42
1540 DATA73,78,32,73,78,80,85,84,0
,583
```

Software Spotlight

Here's your chance to get some help from the experts on how to spend your hard earned pocket money.

Frankie Goes to Hollywood
★ ★ ★ ★ ★
Ocean
£9.95
C64

FRANKIE COMES TO CRICKLEWOOD, or anywhere else for that matter, packaged in a smart box with a free live recording of the band performing their hit song Relax so you can boogie while you plan your next move in the game.

As programs go, this has a lot to offer and the inclusion of games within games reminds me slightly of Terminal's Lazy Jones but this has all the whistles and bells that Jones lacks. It is streets ahead of any opposition and that's not just because it's set amongst the houses of Mundanesville.

The idea is to reach the title screen, a typically contrary Frankie-style idea, to do this you have to visit all the houses in every street of Mundanesville to find objects which will help you to complete the mini games. Really the things people keep in their draws! Old fish, pleasure pills, pistols and video cassettes.

Talking of videos, these play a vital part in the story. Place a cassette in a handy video machine and it reveals a portion of one of the mini games. You are then free to enter the screen. . .but I'm giving away the plot.

Why do you have to enter the mini games? What is the fiendish purpose behind it all? Before you can enter the final screen you must complete your personality. For the purpose of the game, personality is composed of four facets: pleasure, war, love and faith. Each facet is represented by a symbol at the right hand side of the superb 3D graphic screen and each pleasure pill looks like the symbol to which it relates. As the game progresses you gain points in one or more of the



screens for successful results and lose points for failure in the mini games. This is shown by growing and shrinking columns above the relevant symbols. Points are also awarded for various finds in the houses. Occasionally, Frankie will interrupt with a comment which tells you how your personality is developing as a percentage just to cheer you along.

One of the rooms has a locked door which is the entrance to the Corridors of Power forming the Pleasure Dome. This is where the mini games are located and once you have unlocked the door, mapped the maze and fully developed your personality the final door will be found here, but be careful the maze of corridors is full of traps for the unwary.

In one of the houses there has been a murder and once you have discovered the body you are given clues to the murderer's identity and you must not re-enter that room until you have discovered who the felon is. Each time you play the game you will find a different corpse and a new killer.

I will not reveal the nature of the mini games except to say that there are more than 10. Ocean claims that there are 60 problems to solve in all and I believe it. After hours of searching and battling the best rating I achieved was 60% personality. In my opinion this game should rank alongside the best C64 titles and every owner should have it in their collection. **J.G.**

Lands of Havoc
★ ★ ★
Microdeal
£6.95
C64 — joystick optional

A SKULL, SCORPIONS AND SPIDERS, A ghostly graveyard and gorillas, knights in castles and chests are some of the things you will encounter in the Lands of Havoc.

Sador (the player), a reptile in the shape of a man, has come to save the land of Haven from the Dark Lords. His first quest is to seek the Book of Change hidden deep in the village. The Book will reveal the whereabouts of the chest which contains directions to the Dark Lords.

Sador will have to fight and kill to collect the means to protect himself, such as a Talisman from the Mystics and a key from the Garden, before he can enter the Skull in the Wild Wood and destroy the Dark Lords. If he strays into the Wild Wood too soon he will become paralysed – end of game! Only after many hours of play will the final secrets of the Dark Lords be revealed.

This maze type game is of the Sabre Wulf variety. Nine cards map out the maze when laid in sequence (2000 combinations). Half the fun is finding the aim of the game and the other half rearranging the cards in the correct sequence at the beginning of each game (this happens very often until you become experienced). The bottom right-hand corner of the screen has the initials of each card in the pattern they are set out on the table. The redefined character-set makes them almost indecipherable and the pattern is unclear as two of the cards have the same initial.

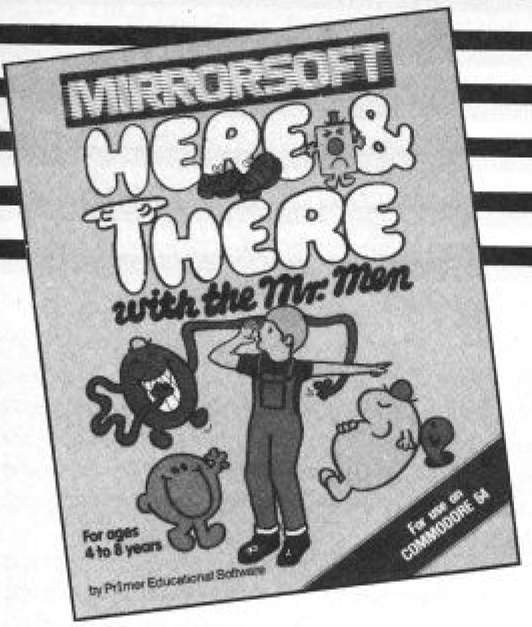
Once defeated the player is informed of his percentage through the game before being included in the Hall of Fame (if he's lucky). The game loads in under three minutes and has high quality graphics. There are eight levels of play, each with an easier/difficult option. The music is excrutiating and should be turned off immediately using the F3 key.

Triangle
★ ★ ★ ★
Argus Press Software
£19.95
C64 — disk only

WITH THE AVAILABILITY OF CHEAP printers, so-called business software is being bought increasingly for home use, and here we have a fully integrated wordprocessor, database and spreadsheet package at an incredibly low price!

The program is very comprehensive and easy to use, although the slender "manual" provided with it would be woefully inadequate to anybody unfamiliar with the processes involved. This is a shame, as the software is unbeatable in terms of pure value for money.

The wordprocessor uses the standard 40 column screen, with words being broken off at the ends of lines as you type. This is corrected at the printing stage, but it does mean that you cannot see the finished article on the screen before



Here and There with the Mr Men

★ ★ ★
Mirrorsoft
£7.95
C64

HERE WE HAVE A SUITE OF FOUR PROGRAMS intended to give four to eight year olds practice in distinguishing left, right, up and down – a concept young children find difficult. An element of route-planning is also involved. The popular Mr Men are used throughout and the graphics, though simple, are very effective.

In *Mr Tickle's Jigsaw Puzzle* the U, D, L and R keys (or the function keys) are used to move doorways in the sides of a box until they are lined up with pieces of a jigsaw puzzle which stand outside. When the alignment is correct the pieces move in and build up a picture of Mr Tickle. The jigsaw pieces are very small and do not line up centrally with the doorways, so the game becomes harder than need be. Children whom I asked to try it soon found it tedious.

Mr Tickle and Mr Grumpy on the other hand, is excellent and quite the best of the four games. Mr Grumpy is inside the box with the same four doorways and Mr Tickle's extending arm needs to reach in to tickle him. To achieve this, the doorways can be moved as before, then Mr Tickle must be programmed with a sequence of instructions for moving his arm along the required route. On later levels, chairs appear in the doorways, adding to the challenge. If preferred, the directions for movement can be given individually.

In *Mr Lazy* the idea of branching to left or right is introduced as you guide a worm up a tree to a red apple. If you are successful he will drop the apple into the mouth of Mr Lazy, who is sleeping below. Another good game.

Mr Men versus Mr Tickle is the poorest of the games in that it is almost impossible to lose! You control four Mr Men, moving on a draughts board, who try to trap Mr Tickle. It could be quite good, but unfortunately Mr Tickle's movements are often rather stupid and he rarely wins.

Taken as a whole, the program is fairly good, though nothing special. It is certainly not up to Mirrorsoft's usual standard of educational games.

P.R.B.

printing. Special commands are accessed logically and easily using the CTRL key, and most of them work very well. A slight problem is that only whole lines of text can be highlighted and moved, but I did not find this a major drawback. Tabs, also, were a bit quirky at times.

Part of the machine's memory is set aside as a "clipboard" or workpage, so that chunks of text can be merged at will, and there is an option for printing globally, so that whole documents may be linked together in sequence. A wide range of printing options is provided, including centring and justification. These are accessed through the familiar "format lines". There are also built-in commands for underlining and italic printing, available on Epson and similar printers, though these are misprinted in the manual.

The database is about the easiest to use that I have seen though it is somewhat limited. The program will not sort records into alphabetical or numerical order. Nor is it possible to scan through an entire file, unless you set aside one of the 17 permitted fields as a dummy, used just to access the records. Nevertheless, as a simple address book or similar database it is very good and has a particularly versatile procedure for print formatting.

The spreadsheet, or financial planning section of the program is excellent, although it is here that the novice would find most difficulty in following the manual. A rather involved system of relative addressing is used when entering formulae, and the explanation given is very thin indeed.

All the sections of the package are fully integrated, so database and spreadsheet files can readily be incorporated into wordprocessor documents. If you run into difficulty, a "help" key is provided, which puts an explanatory window on the screen. The information given, however, usually just duplicates the inadequate text of the manual.

With proper documentation, tutorial and sample files this package would warrant five stars. As it stands, it is still very good, especially when you consider that it is only a quarter of the price of comparable software. Well done, Argus!

P.R.B.

Cave Fighter

★ ★ ★
Bubble Bus
£6.95
C16 — joystick or keyboard

I HAD TO BORROW TWO CASSETTE players before I finally managed to get this game to load – and then only from one side! It's a slow loader (very) so there doesn't seem to be any real reason for this. However, I succeeded in the end, so here goes!

You are deep in the heart of a system of caves, from which you are trying desperately to escape before your energy is exhausted. The caves are infested with bug-eyed monsters, bottomless lakes and pits filled with what look like sharpened stakes, contact with which spells instant death! Fortunately you have nine lives, like a cat, but I guarantee that you will need every one of them.

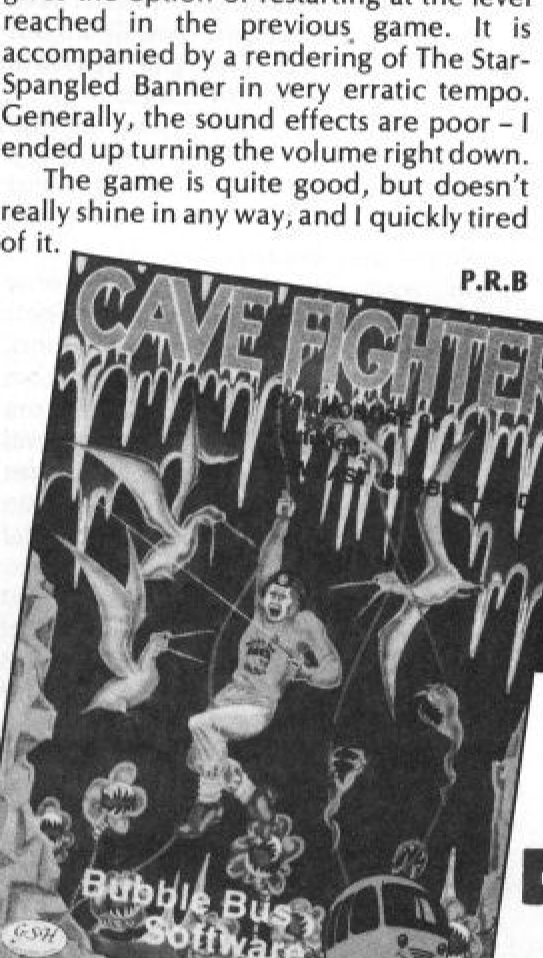
To escape you will need to run, jump, shin up ropes and shoot aliens – in fact it is a fairly standard platform game, with a scrolling screen display. The controls are rather unusual, as you jump only after you release the fire button – the height varies according to the length of time you held the button down. Shooting presents no problems, as you fire automatically in whichever direction you are moving.

There are eight levels, with six caves in each level, and getting through is really challenging!

The graphics are excellent, though the colours are rather muted. Some of the aliens are especially good and move very smoothly, almost like sprites. The title page displays the three highest scores and gives the option of restarting at the level reached in the previous game. It is accompanied by a rendering of The Star-Spangled Banner in very erratic tempo. Generally, the sound effects are poor – I ended up turning the volume right down.

The game is quite good, but doesn't really shine in any way, and I quickly tired of it.

P.R.B



Software Spotlight

Mordon's Quest

★ ★ ★ ★
Melbourne House
£6.95
C64

I'M A "BLAST THE GRIEF OUT OF ANYthing that moves" addict, so when I was asked to review this adventure I thought the intergalactic battle was over. So you can imagine that I loaded *Mordon's Quest* with some trepidation, especially as there wasn't even the slightest hint of any graphics. I was in for a more than pleasant surprise.

Mordon's Quest, by Melbourne House, has been written by Peter (don't you dare give my phone number to anyone) Morland, who also wrote *Classic Adventure*.

It is said by the publishers to be "an extremely complex text adventure". They were dead right about it being text but they were definitely being underhand by using the word "complex". I could think up much more graphic superlatives to describe its twists, turns and generally dastardly tricks and absolutely mind-bending puzzles. As I said, I could be much more graphic in my description, but this is supposed to be a family show!

Like any other text game the computer acts as your eyes, ears and hands and is only limited by the 500 words which it understands; which, when you come to think about it, is an awfully large vocabulary. However, this has been accomplished by using that well known "text compression" system allowing the author to be very descriptive as you travel through the 150 locations which have been cleverly mapped to look like...but that would be telling. Typing in 'help' will point you in the right direction.

The game starts with you, as the possible answer to any Jack-the-lad wizard's prayer, falling out of bed. Your quest is to save the universe, as if you hadn't guessed already.

On your journey you will come across said wizards, matricidal midgets who are heavily into magic mushrooms, and loin cloth clad, human apes, with a strange sense of humour, to name but a few. All you've got to do is collect several parts of a machine which will defeat the young and foolish Bostafer who is bent on immortality at the expense of our total destruction.

All this is delivered in such a way that you could be forgiven for believing that you're actually there. I kid you not, this game with its brilliant descriptions of sequences, events, and locations has got to be a must for any software library, even if you think your forte is "if it moves funk it".

Go out and buy this, you will know it made sense.

D.F.

Tour de France

★ ★ ★ ★
Activision
£9.99
C64 + joystick

SOFTWARE THAT GIVES A LIMITED amount of French lessons is the latest offering from Activision. Based on the Tour de France bicycle race the game boasts some interesting and well planned graphics. I was particularly impressed with the bicycle movements which allow pixel to pixel movement in any direction as the route demands. With each slight movement the sprite appears to change shape accordingly giving an extremely convincing 3D effect, probably the best I have seen.

The game is complex to load because you are given the choice of completing the full Tour or selecting individual stages which are then loaded from data sections on the tape. This means that once you have completed your chosen selection a new game involves reloading the whole program again. Not knowing how the program was written I do not understand why it is not possible to be able to rewind the data section of the tape and reload a new selection. No doubt there is a reason but I would have preferred it to be otherwise.

Another little quibble relates to the fact that although the race can be controlled by the keyboard, the selection of bicycle and country of the player must be made with a joystick. Surely a small amount of extra thought could have released the user from the need to use the joystick at all. I found that the response from the keyboard was better than that from the joystick and that the control of the bicycle was easier.

Having got those gripes off my chest I shall continue with my description of the game. Up to six players can enter the race but each must complete a section in turn. Why the programmer bothered to include the option for a practice mode I do not understand because once practice mode is chosen you must reload the tape to switch to competition mode. I found that by selecting competition mode with six players gave me plenty of time to experiment with different types of cycle and perfect the technique of staying on the road.

To pedal your bike you must press two keys alternately or waggle the joystick back and forth. The energy expended to do this will not give you wrist ache like some sport simulations but it does make you concentrate. Direction is controlled by two separate keys or by moving the joystick left or right with the fire button pressed and you can even select one or two gears as the terrain demands.

French is used throughout the program but most of it can be deduced without any prior knowledge of the language and who knows you may learn a thing or two.

At the beginning of each etape (stage) a screen is given which allows you to select keyboard or joystick operation and it also permits you to select another bike. Each bike has different qualities but it is up to the user to find out what these are.

This means that if you want to change your mode of control or try a new vehicle in mid-race you can do so at the next rest point between etapes.

When I first played this game I thought it was dull but when I discovered the target time for each section on the title screen for each etape, I became engrossed in pedalling like fury avoiding collisions with the grass verge so that I didn't waste time picking myself and my bike up off the floor in an attempt to beat the record time. This added a distinct edge to my one player game and I can see how the multi-player game could become addictive.

At the end of the game you can load a summary screen which displays each player's overall score and gives a breakdown of individual performances. I'd like to say more but it wouldn't load on my version so I'll take Activision's word for it.

J.G.

Nick Faldo Plays the Open

★ ★ ★ ★
Mind Games
£9.99
C64

AMAZE YOUR FRIENDS ON THE 19TH tee with the casually dropped remark, "I hit a 75 at the Royal St George's Club in Sandwich this morning".

Golf addicts and armchair critics alike should enjoy taking up the challenge of one of the most difficult courses in the country.

Using either the keyboard or joystick you can view a detailed plan of each hole so all you need to do is select the right club (with guidance from your caddy!), pick the direction in which you wish the ball to go and choose the appropriate strength for the required distance you wish the ball to travel. Then, adjust for wind direction and speed and hey presto your animated golfer will hit the ball. If all that sounds a bit too easy for you there is



Sword of Destiny

★ ★ ★ ★
Gremlin Graphics
£6.95
C16 or Plus/4 — joystick (optional)

THIS GAME HAS AN UNUSUAL STORY-line in that you start off dead! You play the part of Kelok – not the cornflake manufacturer but the well-known warrior of the Eastern Region. No, you don't work for Anglia Television either!

You have been killed by Xorphas, a warlock of the worst kind, and he has stolen your heart away. "Have a heart!" you said, and he just took it, doubtless to use in some of his unnatural practices! So you are doomed – doomed to wander forever in the Abyss of Death, doomed to the torment of death without peace. Somewhere in the succession of eerie caverns lies your lost heart, which you must regain if you are ever to find rest.

To help you in your quest, you are armed with the famous Sword of Destiny. A strange sword, this, as it hurls bones at people – or perhaps they are thunderbolts. You use it to despatch the vile guardians of death, an assorted bunch of bats, ghosts and grinning skulls, and by sending them packing you replenish your energy a little. Collect any flashing artefacts and you open up further cave sections, but do watch your energy, or astral power level. You only have one life, or rather death, to play with.

To sum up, you must run about and jump from level to level, collecting anything which flashes and bumping off ghosties, ghoulies and things that go bump in the night, trying to find your poor bruised heart. The task isn't easy – it took me several attempts before I even qualified for the high-score table – but at least you get some help from the hot-air flues, which enable you to leap to amazing heights!

Here we have yet another great game from Gremlin Graphics – it's well worth buying.

P.R.B.

Dork's Dilemma

★ ★ ★ ★
Gremlin Graphics
£6.95
C16 or Plus/4 — joystick (optional)

DORKS, AS EVERYONE KNOWS, COME from Dorking, but this one is far from the heart of rural Surrey. His spaceship has crashed on the planet of the Zobwats and is scattered to the four winds, or at least to the 25 mazes. "What is a Zobwat?" I hear you ask. Elementary, my dear Watson – it is nothing more than a Tawboz spelt backwards, but it can take many forms, all of them nasty!

If you are wondering whatever I am dorking about, let me explain. The aim of the game is to collect a piece of spaceship from each of the 25 chambers, then reassemble them. To collect these components, all you need to do is to kill a minimum of 10 Zobwats in each chamber. The method of killing is easy. You drop a bomb when they are close to you, then you dork out of the way pretty sharpish! The trouble with Zobwats is that they tend to surround you in a corner, so you can't get away after dropping a bomb. You have four lives, however, so it isn't as bad as it sounds. Some of the chambers are rather cramped, making it difficult to manoeuvre, and you start in a different room each time you play, so it is hard to work out a technique for dealing with each one.

Gremlin Graphics has produced some really excellent games for the C16 and Plus/4, and this is no exception. Complete with high score table, joystick or keyboard options, sound on or off, it is a thoroughly professional piece of programming. The graphics are colourful and varied, and the sound effects are just right.

The game is entertaining, and challenging enough to maintain interest. One word of advice – don't rush around too much, but be patient and you can achieve a very high score.

P.R.B.

---

also a 'fine tune' which allows you to make last minute adjustments to the strength and direction of the shot as your golfer swings his club through the air.

You can then watch the ball fly across the screen and land on the fairway, or the rough, or the bunker, or the water hazards or even go out of bounds! The usual everyday frustrations of the handicap golfer prevail. At least you have the consolation that if you miss a two foot putt no one else is watching.

The weekend hackers and club handicap players alike will relish the chance of pitting their wits against an open championship course. Even those of you with no golfing experience should quickly learn enough of the rudiments of the game to play many enjoyable hours trying to hit a small white ball into a small white hole. This game is easy to learn but, like real golf, it's difficult to master. Holes in one, eagles, birdies and pars are all attainable but beware – bogey's abound in plentiful supply for the unwary.

P.T.

Thing on a Spring

★ ★ ★ ★
Gremlin Graphics
£7.95
C64 + joystick

IT'S A LONG TIME SINCE WE HAVE SEEN any C64 games from the stable of such classic games as Monty Mole and Potty Pigeon. However, Gremlin Graphics has now returned with vengeance with what is sure to be its latest chart topper – Thing on a Spring.

As ever your mission is to rid the World of some evil that has been unleashed upon it. This time, however, it's not the usual mad scientist or the Warsaw pact that you're out to stop, it's an evil goblin who is trying to rid the world of all its treasures.

This time you don't play the role of a tank commander or a superhero instead you find yourself quite simply as a Thing on a Spring.

Platform games have certainly become as common as Space Invaders, and it's quite a surprise that Gremlin Graphics has released yet another one onto the market. Don't worry though, this isn't your run of the mill platform game it's far superior to most that have been launched recently.

Controlling Thing is no easy task, being a spring he tends to bounce around quite a lot, if you're not careful you'll find him bouncing right into the clutches of some evil monster who will reduce the amount of oil on his spring. Too little oil and guess what happens!

Hidden around the numerous rooms in the Goblin's lair are five switches. Thing needs to collect these before he can get around all of the complex. Finding the switches and finding out what they do turns what could have been a normal platform game into a real arcade adventure. It took me an hour to get out of the first room.

Thing on a Spring has some superb graphics and sound. Gremlin's claim that Thing is 'cute and everso lovable' isn't that far from the truth.

Rush out and buy it, you'll love it!

S.C.

Software Spotlight

Whirlinurd

★ ★ ★
US Gold
£9.95
C64 + Joystick

WHIRLINURD HAS A PASSION FOR collecting things. TV sets, grapes, toy yachts or candelabra have such a strong pull on his acquisitive nature that he will risk life and limb to reach them.

Our hero lives in a dangerous world of mazes which are guarded by snakes and small round creatures with antennae. Whirlinurd must collect as many objects as possible avoiding contact with these creatures and can fly by using a rotor blade fitted to his head.

The first few mazes are fairly straight forward but the complexity increases as you move from screen to screen. Eventually, you reach mazes in which you must find keys which fit in locks of the same colour. Placing a key in a lock opens up another part of the maze which contains one of the sought after objects but also releases another creature to add to your problems.

I found this game quite challenging to play but I can't say that it was more exciting than the host of other games in the same vein. Like most of these games, once you work out a pattern to follow each maze can be traversed without a great deal of difficulty. One good feature is the ability to start at a higher level, preventing the need to start from the beginning every time.

J.G.

A View To A Kill

★ ★
Domark
£10.99
C64



SOMETIMES PROGRAMMERS CAN GO overboard with special effects at the expense of what could otherwise be a superb game. A View to a Kill is one such example.

Based closely on the plot of the film, the game is split into three parts, or four if you count the title sequence which merely computerises the familiar opening sequence to all the Bond movies.

The first scene picks up the story where Bond is chasing the hang-gliding villainess May Day. He has requisitioned a taxi for this purpose and he must avoid collisions with the buildings and the mad motorists of Paris if he is to intercept May when she touches down. To help, you are faced with a rather complex screen which shows a 3D view from the front window of the car, a bird's eye view of Paris and a gauge which tells you how far away May is.

Although this is an excellent piece of programming, I would prefer a larger view of Paris so that I could see instantly where the hang glider is. Control of the car whilst looking at all the other displays is virtually impossible and you can't tell if the street you are on is a cul-de-sac or if you will end up driving yourself in Seine.

The next two parts are action adventure games. The first is set in the San Francisco City Hall and you must rescue the beautiful Stacey from the lift shaft where arch-villain Max Zorin has trapped you both after setting fire to the building. As the fire creeps from room to room you must try to obtain the necessary equipment not only to rescue Stacey but able to use to escape from the building. Using a 'duck shoot' menu you can examine or use the objects which you find, command Stacey to follow or wait behind and if you go desperately wrong you can quit and start again.

The final part is set in the mine in which Zorin has placed the device which will soon explode to make sure that the occupants of Silicon Valley have had their chips. Before you can defuse the bomb you must search the mine for suitable gear so that May Day can help winch you down to do your work. This section is very similar to the previous one, though in my opinion it is the better of the two.

Each section of the game relies on success at the previous section to enhance your chance of being able to complete it. For example, a geiger counter is hidden in the City Hall which will help to locate the bomb in the mine. This gives a theme by which the three parts hang together as a whole but it also detracts from the full enjoyment of the individual parts.

My advice is to spend your money on Bond but make sure that it is a Premium Bond, you have more chance of success and it may prove more rewarding.

J.G.

Beer Belly Bert's Booze Biz

★ ★
US Gold
£9.95
C64

ALLITERATIONS ARTFUL AID ASSISTS another attempt at aggregating akkers for US Gold. Unfortunately, I don't think that it will, not because it is a bad game but because it offers nothing more than the majority of ladders and platform games currently available.

After a rather crude (by US Gold standards) loading sequence the title screen gives way to a view outside the brewery where we find our pot-bellied hero standing outside three doors. The doors are marked Control, Production and Shipping but as the game is similar no matter which door you enter we shan't linger on the significance of these labels.

On entering one of the rooms you are faced with a scene of part of the brewing factory. There are doors on most of the levels and one or two of the floors have exit signs. Somewhere on the screen is a key which unlocks the way to another room and it is Bert's aim to grab the key and move on.

Ventilation ducts blast air upwards and if Bert stands on one of these he is carried by the draft up on to the next floor. To escape from some of the rooms the air blast provides the only pathway to the exit.

Occasionally one of the doors opens and a mean little character appears. He walks up and down like a security guard and will fire the odd pot shot at Bert who must dodge or die. Bert is also armed and he can retaliate with a well-aimed shot which will kill his opponent who then disappears only to re-appear from behind another door.

Some of the rooms are linked and the only possible means of escape is to enter the other room and face its perils only to reappear on the desired level of the original room to make his escape.

For my money, this is not as good as the Bounty Bob games which US Gold markets but as you go from room to room the challenges increase in number and variety and I found that it at least held my interest.

J.G.

This month Runecaster delves into the secrets of Exodus: Ultima III and encounters elves, dwarves bobbits and fuzzies!

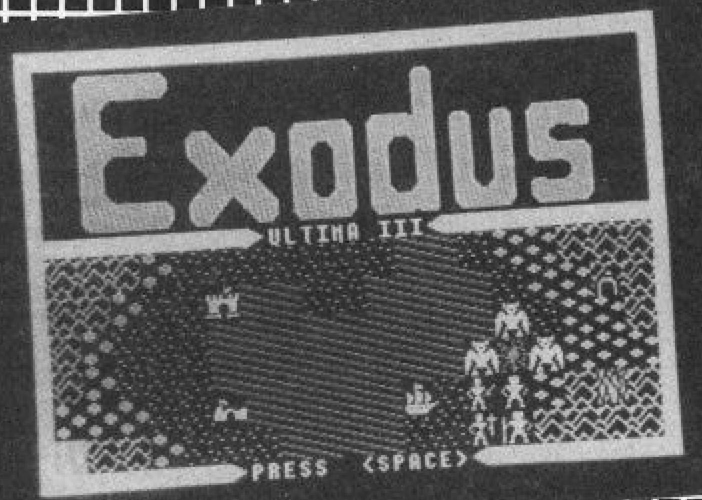


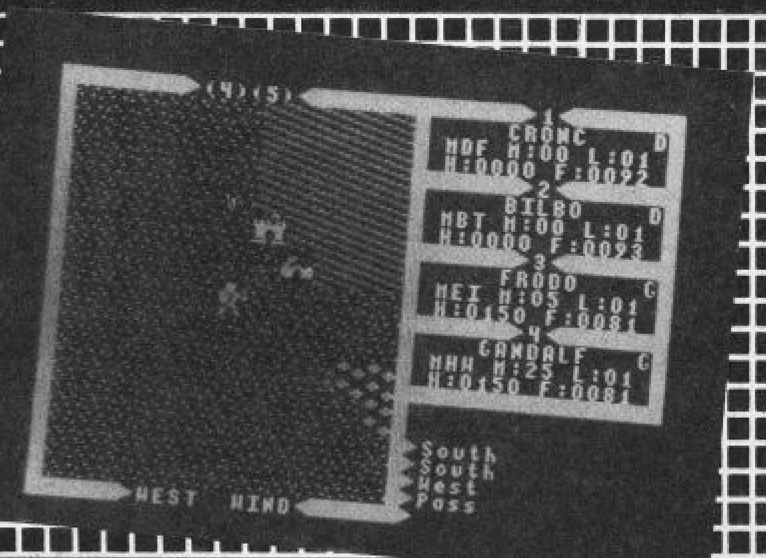
Oh boy...oh boy!

ARE YOU A FRUSTRATED BARBARIAN with your sword rusting on the wall? Perhaps, a mace bearing cleric resting between quests. Did you want to play *Dungeons and Dragons*...but couldn't find anybody else to join you at the right time or place – and, for the hours (days!) necessary for a worthwhile quest?

Get down that sword, oil the hiking boots, get in a goodly store of victuals and prepare for an extended period away from the daily rat-race.

If you haven't got a C64 complete with disk drive, now is the time to invest...and, while you're at it, get a copy of *Exodus: Ultima III* by Origin Systems Inc., distributed in the UK by US Gold. This is such an interesting find that for the first time I shall be looking at only this one subject. I hope you like role playing games!





In the Past

Some four years ago there was *The Valley*, where you could choose your character type (fighter, cleric, wizard etc.) and set off adventuring — battling a host of monsters, finding treasure and casting the odd spell. At the time it was quite a hit but with limited graphics and designed for the 16K PET, it now looks a little pale!

There have been several attempts to bring genuine 'free flowing ' role playing games to the C64, notably perhaps the *Dungeonquest* series which included *The Temple of Aphsai*. They have been partially successful but have nearly all been for single adventurers (no teams) or have not really had that spark of 'alternative reality' that makes for total player involvement.

Presently Available

Other computers, especially in America, have had good role playing games written for them, *Wizardry* for the Apple is one that immediately springs to mind. Now, at long last C64 users in the UK have their turn!

Ultima III could well prove to be a game which adventurers will spend many thousands of hours playing. Sadly, it is disk based only, with no real hope of a cassette version ever appearing.

The operating system for the adventure comes on one side of the disk, with a program for creating a separate 'scenario disk' on the other side. Before you can venture forth you must create this separate scenario disk.

This creation routine takes some time (about 12 minutes!). When complete you have to switch off the computer (and disk drive) and reload the operating system before you can play. The sole difference in 'playing' and 'creating', is whether you press the SPACE bar or 'C', once the main titles appear. Don't make a mistake, as even loading the operating system takes several minutes!

This isn't a game to be played lightly. It takes time to set up and even more time to get over the initial learning period – what keys do what, which is the best way to make up your team of adventurers (you can control up to four in a party).

For this reason I could suggest a little cheating to start with even though you will be fretting to get going. Make up two scenario disks! Make sure you format the disks with exactly the same 'disk name' and I.D.

We are now almost ready to start! Load the operating system...up come the main titles...press the SPACE Bar...whirr, whirr: not long to wait now...insert your scenario disk when told to...we're there!

The initial screen displays a moving vignette of a team travelling around in Sosaria fighting monsters and generally

enjoying themselves. You could watch this for longer. . .but what about your own team?

Pressing SPACE gives you three options: Return to the moving vignette; organize a party; journey onward. You cannot choose the third without first creating your characters and forming a party so, your initial choice is determined for you.

Again a menu, giving the option to examine the main characteristics of existing characters, creating new characters, forming or dispersing a party, removing (terminating) characters and finally returning to the previous menu.

Vital Statistics

Initially you must create a selection of characters – at least four. You can go adventuring with less than four but this is not recommended, especially to begin with!

You may have up to 20 characters 'in store' and in creating each one, you must define their roster number (1-20), name, sex, race (human, elf, dwarf, bobbit or fuzzy). Each racial type has different potential abilities – dwarves are strong, elves are dexterous etc. You must bear this in mind – you wouldn't want a clumsy thief would you!

Next, you have to choose the character type – barbarian, thief, wizard, ranger etc. There are 11 possibilities given and this choice must be made carefully.

Not only do you want a balanced Party that can deal with whatever it meets along the way but you must also allocate points (up to a total of 50) to your character's strength, dexterity, intelligence and wisdom. This, together with race and type, will modify a character's capabilities.

The permutations possible are quite enormous but after a few forays into the unknown you will start to get some idea of a viable Party's requirements. You will probably 'zero in' on a toughie of some sort – preferably one that can also use a little magic.

Then you'll need someone who can use prayers (cleric, druid etc), if only because they can heal injured friends and also magically examine treasure chests for traps. Thieves seem to be very useful – personally I like the female, elven variety! Then perhaps the pure wizard type or even your original red blooded barbarian.

Study the table that gives the characteristics for each profession, this will tell you what weapons, armour and spell type each may use and what compromise is made (if any) for the use of magic.

Make notes of each character's attributes as you type them in. This is quite important, because once created but before setting forth, you cannot check such attribute points as strength, dexterity, intelligence and wisdom.

You may always call up a character's status during play but once the adventure proper has started you cannot return to the creation stage. When you have created sufficient characters then 'Form a Party', entering the four characters you have chosen, then return to the main menu and Journey Onward.

Cheat!

If you want to 'cheat a little' press 'Q', which allows you to save your position, remove the scenario disk and switch everything off! Now, reload the program operating system but this time put in the second scenario disk you created.

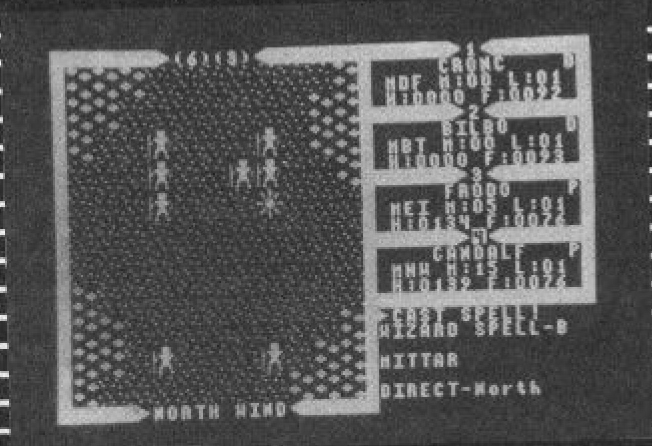
Go through the character creation and party formation routines, giving exactly the same information as you did the first time. The scenario disk records all your characters' changing data as the game progresses. . .including their death.

As this is quite likely to happen fairly often to start with one of your cheat disks should be kept as 'clean' as possible. Use this one when you start each time and 'save your position' to the 'clean' one as your characters gradually improve their standing.

It is almost impossible to get away without somewhat corrupting these disks as you continue but at least it will eliminate a fair amount of time being spent creating characters when you would rather be learning how to play!

At Large in Sosaria

You are there at last, with the wind in your hair, standing with open rolling plains behind you and a small walled city in front of you. The display shows a plan view, with one character indicating the position of your party. water is shown to the right of the map and two buildings represent a castle and the small town. You can see some woods to the west.



To the right of the map is a 'vital signs' display of your four characters. This gives you at a glance, such information as their food remaining, hit points (0=death), magic points (spell casting ability) and their Level (all start at Level 1).

Further status information can be had by typing 'Z' (ztatus!), this will tell you how much gold they have, what weapons or artefacts they are carrying and their present strength, dexterity etc.

Naked into the World. . .

Each character starts with a dagger and cloth armour (!) but you must tell them to 'R'eady the dagger for use and 'W'ear the cloth armour. . .otherwise they will enter battle using their bare hands and in their skins!

They are also provided with 200 pieces of gold. The sensible thing to do next is 'E'nter the town and visit either the weapons shop or the armoury. Weapons may be bought and sold at the listed price (no haggling!) as also may be food when you need it, from the grocers.

Just a Quick One. . .

There is also a pub where useful (?) gossip may be heard (bought!). In some towns you may also find other useful places, but you always need to search for these very carefully. You can even talk to the various people you meet in the streets.

You are now kitted out to the limit of your purses and leave the safety of the town. You have of course read the instructions that came with this gateway to another world but you won't have absorbed all they had to say, and will perhaps have noticed just some of the omissions – like, there is a list of 32 spells. . .but only one is explained!

The spells have intriguing names such as REPOND, MITTAR, PONTORI, SANCTU, LORUM etc. Use of these depends upon the number of magic points a character has, these in turn depend upon character type and how you allocated those original 50 points. Only by playing, can you begin to understand how to achieve a workable balance.

Spell That Again

Try the spells at various times – whilst walking along, in battle, lost underground etc. See if you can work out what they do (I don't know, I've only been playing for about 30 hours so far!) MITTAR and

SANCTU are both very useful and may be cast easily by Level 1 characters.

When you meet any other being outside in the wilds, it (or they) are naturally bloodthirsty (yours!) and aggressive. It is occasionally possible to evade them but...

When combat occurs, the display shows a larger scale map, with the four members of your party towards the bottom and the evil one(s) towards the top. Once conflict has begun there is no retreat...and the losers are dead!

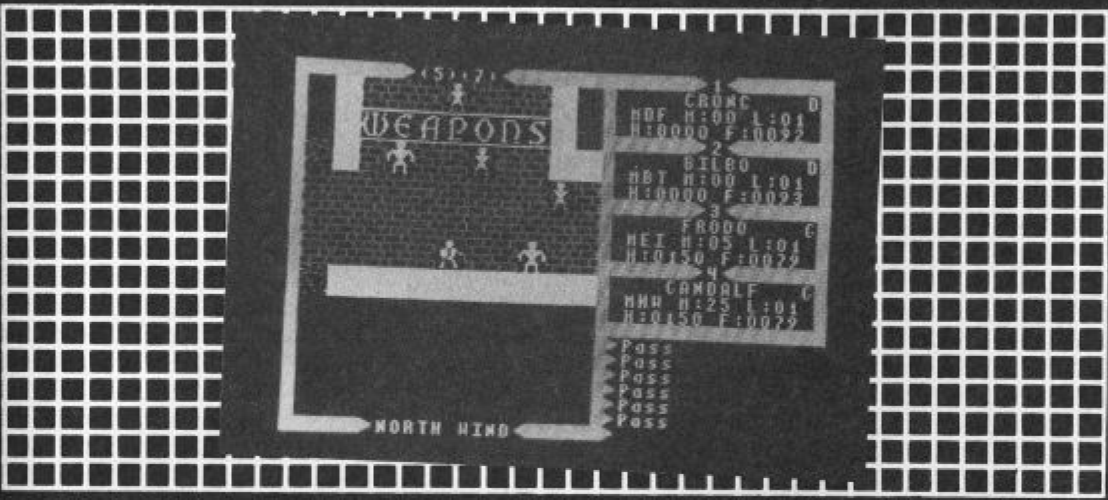
Battle at a distance is possible with a bow (unlimited arrows...great!), magic or even by throwing a dagger. Usually in the early stages, hand to hand combat takes place. Your team may swing their swords, maces, hands etc. only to the north, south, west or east. The enemy may also strike on the diagonal.

be in a hurry to open the chest...a spot of 'healing' (also performed by your clerical type) may be worthwhile – the baddies have a habit of appearing at quite the most inopportune moments!

Monsters come in a variety of different guises, from renegade wizards, orcs, trolls, ghouls, zombies, giants, bradles to daemons, griffins, dragons, devils, balrons and even sea serpents, to mention but a few!

A Life on the Ocean

There are also pirates who can fire at you from their ships at sea. If you happen to be at sea (yes, your party can board and sail the occasional ship), there is even ship to ship combat. If you are at sea then watch the way the wind is blowing – you cannot sail into wind, you must tack!



Each time a member of the team is hit their 'hit points' decrease. If they go to zero, they're dead. Once over the initial stages of 'finding out what to do', I found the combat and magic 'balance' one of the best I have come across.

Every time a member of your party kills one of the monsters, points are added to his or her 'experience points'. The greater the number of these, the higher the 'Level' the character becomes and the more effective that person is at everything attempted.

No Clock Watching

There is a 'real time' element throughout the game that is acceptable and although you must act fairly quickly, you do not have to be crazy fast as in some games.

Having won your victory, the vanquished foe leaves behind the proverbial treasure chest. Someone has to open it to collect the spoils. Needless to say they are often booby-trapped with a variety of unpleasant surprises.

If you have a clerical type amongst your party then all is solved by invoking APPAR UNEM. If this spell (prayer?) fails then it is worth trying again. No priestly type – then the next best thing is a thief, they can sometimes (!) spot a trap.

If it has been a costly battle then do not

After you have suffered a loss of hit-points during combat, they will gradually come back as time passes. The same goes for any magic points extended in casting spells. There are also said to be places of healing in remote areas with restorative powers which may be purchased – at a price.

There are any number of towns dotted around the lands of Sosaria but most are not laid out in such a simple fashion as the one from which we started! In fact some are pure and simple mazes...so keep pencil and paper handy.

This really goes for the whole game, as, although you think you know where you are, all is not quite as clear as it seems. There are many, many times when the terrain precludes you seeing much further than a few metres – in mountains or thick woods for instance. There is even one town I've come across like that!

In the States you can buy *The Secrets of Sosaria* which maps towns, castles, dungeons and magic realms. It also includes other useful information. Where can I get a copy?

Be careful what you 'E'nter. Dungeons are shown as a 3-D maze, the four keys you've been using as 'move N,S,W,E' now mean forward, retreat and turn left or right. The problem with dungeons is that they are dark!

Let There be Light

It is possible to light your way magically but the spell lasts only a short time and can only be considered as a stop-gap measure. You have to find/buy (?) a lamp or torch.

There are also a few hidden 'Moon-Gates'. Travelling through these will teleport your party "to many glens of the old knowledge". These 'portals' are affected by the twin moons of Sosaria – Trammel and Felucca – and to help you in your understanding of the working of these, the phases of the moons are included in the main display!

To stop playing, press 'Q', to save your Party's present status and simply remove the disk – but only after disk drive has stopped running of course – and switch everything off.

On your next visit to Sosaria, you may either carry on where you left off, or disband the party and form another one. New characters may also be introduced at this time perhaps to be taken under the wing of more experienced adventurers!

This program is really the most complex and rewarding I've been fortunate enough to play for ages. It even includes the option for regular adventure verb commands in special situations, such as 'KISS PRINCESS' or 'JUMP CLIFF'! In fact there is a lot more to this game than can possibly be covered here.

There is an underlying plot to *Ultima III*. In addition to being a first class role playing adventure. It is concerned with events that took place in a previous adventure: *Siege Perilous*. Any knowledge of this is completely unnecessary, it merely sets the scene for this vastly uprated descendant.

The basic story-line is the rising power of evil and the call by Lord British to would-be adventurers, to rid the land of an as yet unspecified malignant evil entity. There are some clues that may or may not help one of which is the word Exodus scrawled in blood, found on a derelict merchant ship.

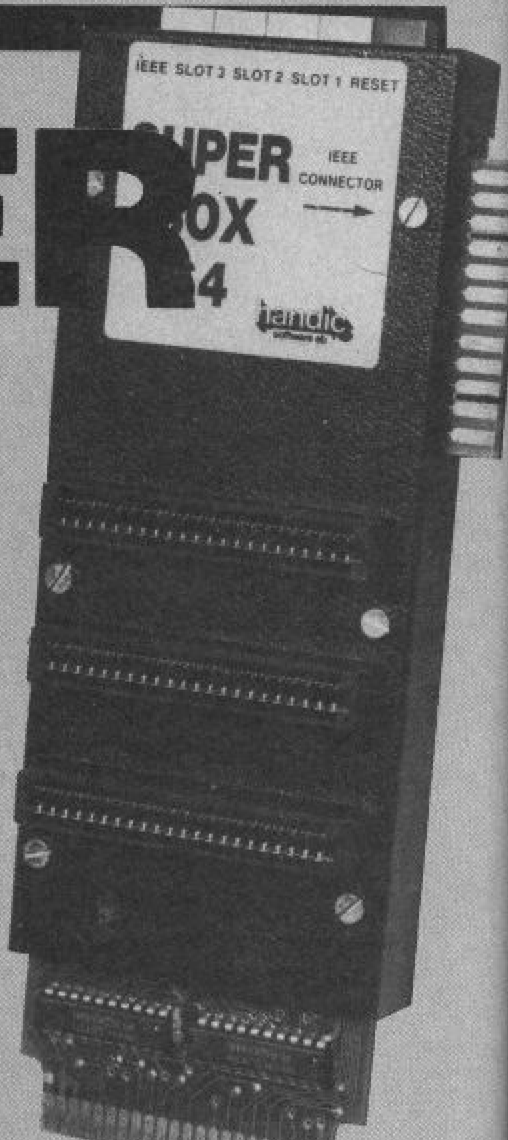
Perhaps I might get round to the deeper meanings of *Ultima III* in a month or six, but for now my rusty sword has been cleaned and honed to perfection, my boots have new nails and I've just had my last cigarette. Be seeing you...in Sosaria.

Foot Note

Unfortunately, US gold, the UK distributor, has only included in the package the game disk, the book of play and the reference card – omitting the map and additional booklets describing the spells available to the clerics and mages. Since these will be vital to your success in the game, Computer Gamer will be publishing the vital mission material in its September issue.

Barry Miles has been looking at Handic's box of tricks and he found one or two surprises.

SUPER BOX 64



THIS PRODUCT IS A VERY INTERESTING combination for the C64 user. It is both a motherboard to enable up to three cartridges to be installed simultaneously and also an IEEE interface.

If you own an earlier Commodore machine you may already own a disk drive. The early models used the IEEE 488 parallel bus normally known as the "I-triple-E" bus. This was a slight variation on the original design by Hewlett Packard.

When the Vic 20 arrived. Commodore, to many peoples amazement, changed to a Serial system. This resulted in data being passed along the line with the bits following one another rather than eight bits going in parallel. Obviously, this slowed down the data transfer rate and the 1541 disk drive used on the Vic 20 and the 64 has been slated for its slow performance ever since. This same process has been adopted on the C16 and the Plus/4.

The faithful Commodore user must buy some form of interface to make the 4040, 8050, 8250, 1001, and 2031 drives work with the 64. Oxford Computer Systems Interpod – happily still available – solves this problem but enables these disks to communicate with the 64 at only the same speed at the 1541.

What the average user wants is to be able to take advantage of the maximum speed of which the disk drive is capable.

Generally speaking the interfaces which are available rely on placing some machine code in the memory of the 64. This sometimes conflicts with memory demands made by advanced packages.

There are difficulties when it comes to connecting a number of cartridges to the 64 and switching between them. Many and more ingenious manufacturers are putting protection into their cartridges which creates problems.

Handic's approach was to supply an IEEE Interface board enabling the user to choose between three cartridges plugged in at once, whilst at the same time being able to use Commodore disk drives at full speed. In addition, the product has a Reset button.

This was an interesting aspect of the design because it combined three desirable features. It is frustrating to plug in another board in order to use your cartridges only to find that you cannot use your disk drives.

The Superbox is a Swedish product and seems very durable. The rigid metal case looks as if you could drive a car over it without damage.

The only protrusions are the edge connectors. One for your IEEE cable and the other for plugging into the cartridge slot on the machine. There are rubber feet on the bottom to make sure that the box will lie flat. No RAM in the 64 is used and this avoids any memory conflict.

In order to pick up the necessary signals it is necessary for you to connect a crocodile clip to a single resistor on the C64 board. This is not a difficult task to do but nervous and ham-fisted users may prefer to get their dealer to do this for them. Fortunately you can remove the box, should you need to do so, whilst retaining the connection.

The switching has been very pleasantly designed indeed. At the far end of the box, is a set of five switches. As you switch on one cartridge slot the other switches pop up: the only way you can accidentally have two cartridges in the circuit at once is by pushing two buttons simultaneously.

The IEEE button will connect you to your parallel IEEE device. The Reset button is a great advantage because the 64 unlike its successors, the C16 and the Plus/4, had no reset button whatsoever. By pressing this red reset switch a "Cold start" is performed which leaves the contents in RAM unaffected, if the system has "hung" use of this button may get you out of trouble without losing any data.

In Use

Testing the box with a variety of cartridges showed that most of them would work perfectly OK. Certain types of cartridge however would not function correctly.

There is a somewhat startling warning in the brief A4 sheet manual saying that if an IEEE unit is connected to the board and not switched on when the machine is running, the Superbox 64 may be damaged. Inquiries of Handic UK, confirmed that this was merely an example of Swedish caution and not to be taken too seriously.

You are warned not to connect more than two IEEE units at the same time and to make sure that all units connected are switched on before running the system. I was not tempted to test this out with a variety of extra IEEE units!

Conclusion

At £69, the superbox 64 is not a cheap product in fact if you compare it directly with another mother board you will find it extremely highly priced. However you are really getting two products for the price of one! Highly recommended.

The only difficulty likely to be experienced in using the Superbox arises because of its all-or-nothing approach to life. You are expected to be using either all Serial or all Parallel IEEE devices. An addendum to the manual gives a partial fix for this.

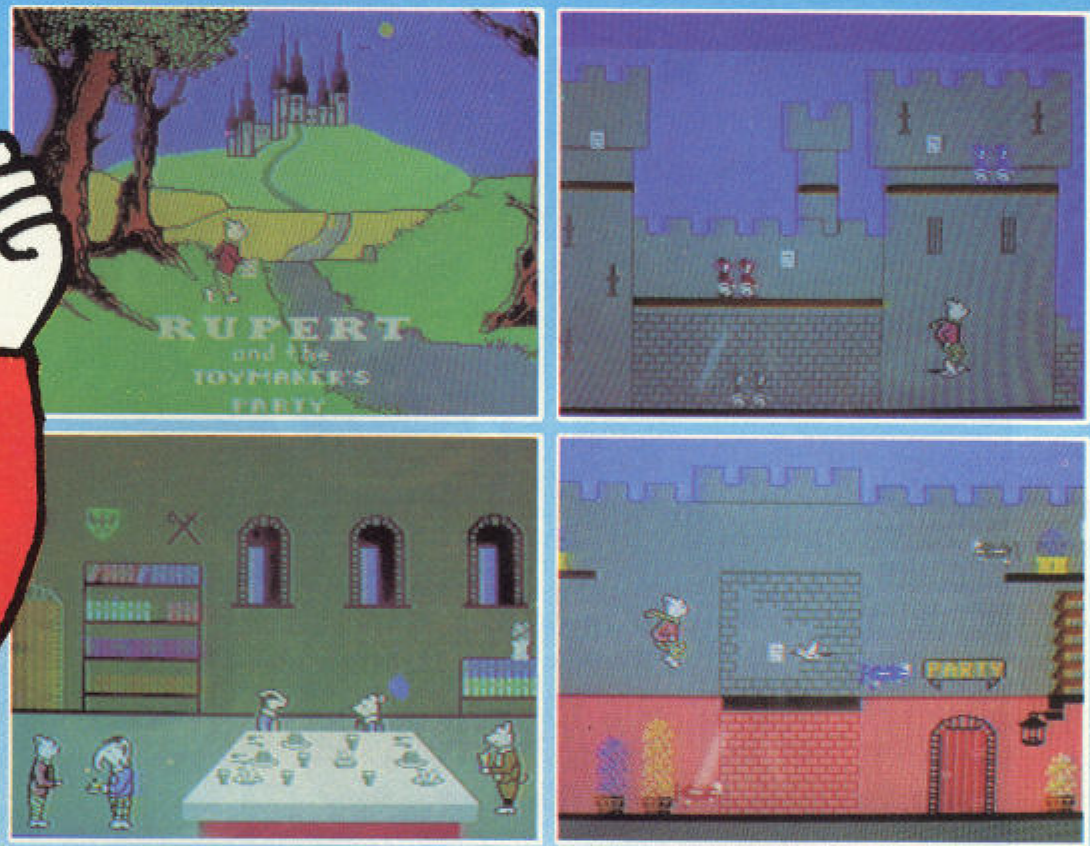
By PEEKing five memory locations and recording their content manually, you enable the use of the Reset button without the loss of a Basic program. Thus, you press the button to switch between IEEE Serial and Parallel in either direction and then press the Reset button. Finally, you POKE the data which you have recorded into the same memory location.

RUPERT

and the

TOYMAKER'S PARTY





Jolly old Rupert can walk, jump, leap, climb stairs – and even fly – but can you get him to the party?

32 challenging screens of animation and fun.

QUICKSILVA

The most powerful games in the Universe

Spectrum 48K
CBM64

Quicksilva Ltd, Liberty House,
222 Regent Street, London WIR 7DB.
Telephone 01-439 0666

£7.99

OR... MADE BY ALIEN



THE FINAL CARTRIDGE

FOR CBM **64**



£ 50,-

- **A new outside OPERATING SYSTEM** built in a cartridge, that does not use any memory and is always there,compatible with 99.9% of the 20.000 C64 programs
- **24K EXTRA RAM** for basic programs.
- **A DISK TURBO** that not only **LOADS 6 TIMES FASTER** but also **SAVES 6 TIMES FASTER!!**
- **A PRINTER INTERFACE** that is compatible with all the well known centronics printers and Commodore printer programs. Prints out all the Commodore graphic characters and control codes.
- **A SCREEN DUMP** for normal and hi-res sreens (c.q. Doodle). With a special built in routine that searches for the memory adress of the hi-res picture automatically.
- **PREPROGRAMMED FUNCTION KEYS.**
  Run, list, load, save, catalog etc.
- **MACHINE LANGUAGE MONITOR.** Very sophisticated assembler/disassembler/moni-tor
- **BASIC 4.0 and DOS COMMANDS.** Dload, DSave, Catalog: reads directory without disturbing your programs.
  BASIC TOOLKIT. auto, renumber (incl. goto) help etc.
  RESET SWITCH. Resets to monitor. Resets with old (leaves your basic program intact) Resets every protected program
- **CASSETTE TURBO** that loads and saves programs **10 TIMES FASTER** including files and records.

H&P COMPUTERS

Wolphaertsbocht 236
3083 MV Rotterdam.
Netherlands
Tel.: 01031-10231982
Telex: 26401 a intx nl

Mastering Machine Code

A.P. and D.J. Stephenson explain the use of code other than decimal and hexadecimal.

PART 1 OF THIS SERIES DEALT ONLY with decimal and hexadecimal codes because they are the most popular for general purpose use. However, there is another method of representing binary known as Binary Coded Decimal, (BCD for short). The 6502 microprocessor is particularly well equipped for handling this type of code. The common arithmetic processes of addition and subtraction are, by default, carried out in normal two's complement binary. It is possible to change this by using the instruction SED which is the mnemonic for SEt Decimal mode. Once the machine has executed SED, (op-code F8 hex) all subsequent arithmetic is carried out in BCD until cancelled by CLD (op-code D8) which is the mnemonic for CLear Decimal mode.

The BCD code

Before considering the areas where BCD might be useful, we must understand the mechanism of the underlying code. Perhaps the easiest way to start is by examining the following byte, artificially split into two separate nibbles for ease of understanding:

1001 0010

If this was interpreted in terms of absolute (unsigned) binary, it would represent 146 in decimal, 2+16+128. If interpreted in signed binary (two's complement) it would represent -110 in decimal. Finally, if interpreted in BCD, it would represent the decimal number 92. In fact, the space we left between the two nibbles, although artificial, turns out to be a direct aid to the understanding of the mechanism behind BCD. Each nibble should be thought of as a separate black box, the right-hand box represents the units column and the left-hand box represents the tens column. Here are some BCD examples:

1000 0111 = 87.
0110 0001 = 61.
0111 0111 = 77.
0101 0000 = 50.

Redundancies

Apart from the difference in interpretation, the BCD code has, what the code boffins are pleased to call, redundancies. We can see what this means when we consider the number of possible ways in which 8 binary bits can be arranged. For simplicity, let's start with two bits. These can only be arranged in four ways, 00,01,10 and 11. Three bits can be arranged in 8 ways, 000,001,010,011,100, 101,110 and 111. So for two bits, there are four ways (2^2) and for three bits there are eight ways (2^3) so we can deduce from this that the general formula for knowing the number of ways of arranging N bits is 2N. Now, in the case of an eight bit byte, we shall expect that there are 2^8 = 128 ways of arranging the bits. Normal binary and hexadecimal codes use all these combinations but BCD doesn't. In fact, as the following table shows, there are six illegal combinations (redundancies) in each nibble:

BCD	Decimal		
0000	0	1010	
0001	1	1011	
0010	2	1100	illegal
0011	3	1101	combinations
0100	4	1110	
0101	5	1111	
0110	6		
0111	7		
1000	8		
1001	9		

Program 12.1. Convert BCD digit to ASCII

```
10 033C         ! CONVERT ASCII TO BCD DIGIT
20 C000         *=$C000
30 C000         MEM        =   $FB
40 C000 38                 SEC
50 C001 A5FB               LDA MEM
60 C003 E930               SBC #$30
70 C005 85FB               STA MEM
80 C007 60                 RTS
```

Program 12.2. Converting ASCII to BCD digit.

```
10 033C         ! CONVERT BCD DIGIT TO ASCII
20 C000         *=$C000
30 C000         MEM        =   $FB
40 C000 18                 CLC
50 C001 A5FB               LDA MEM
60 C003 6930               ADC #$30
70 C005 85FB               STA MEM
80 C007 60                 RTS
```

As you can see, the illegal combinations are those greater than 1001. These are the combinations normally represented by the letters A to F in hexadecimal.

We conclude therefore that representing numbers in BCD is inefficient because, out of 16 possible combinations, only 10 are used. In percentage terms, this represents an efficiency of only (10/16) × 100 = 62.5%. As far as memory usage is concerned, we need about 30% extra memory space to store numbers of BCD form because a full nibble is still required for each BCD digit in spite of the combinations which remain unused. We can represent efficiency in a more obvious way by comparing BCD with unsigned binary in respect of the largest number capability. The largest legal combination in BCD is 1001 1001, 99 in decimal, whereas in unsigned binary the largest is 255 decimal.

BCD addition

Since we mustn't use combinations greater than 1001, what exactly happens when we add 1 to 9 in BCD? Under normal binary conditions, the addition would be quite straightforward:

```
      0000 1001
add   0000 0001
      ---------
      0000 1010
```

But, the right hand nibble now has an illegal combination! However, if the 6502 had been previously instructed (by means of SED) to perform the addition in BCD, it would perform as follows:

```
      0000 1001
add   0000 0001
      ---------
      0001 0000
```

Note that adding the 1 has resulted in a premature carry from the right-hand nibble into the left-hand nibble. It is called the 'half carry' to distinguish it from the normal carry out from the msb end. Note carefully that the total (10 in decimal) is correct in BCD format. How does the microprocessor perform this bit of trickery? Actually, the answer is simple. It performs the addition in normal binary and then tests the result. If the right-hand nibble is illegal it then adds a further six. This may seem mysterious but is quite logical when you think about it because there are six illegal combinations which must be skipped. Examine the following example, using the previous figures:

```
        0000 1001
add     0000 0001
        0000 1010   illegal, so

add 6   0000 0110
        0001 0000   result correct in BCD
```

Program 12.3 Convert hex digit to ASCII.

```
 10 033C              ! CONVERT HEX DIGIT TO ASCII
 20 C000              *=$C000
 30 C000              MEM           =   $FB
 40 C000 18                         CLC
 50 C001 A5FB                       LDA MEM
 60 C003 C90A                       CMP #10
 70 C005 9002                       BCC OVER
 80 C007 6906                       ADC #6
 90 C009 6930         OVER          ADC #$30
100 C00B 85FB                       STA MEM
110 C00D 60                         RTS
```

Program 12.4. Convert ASCII to hex digit.

```
 10 033C              ! CONVERT ASCII TO HEX DIGIT
 20 C000              *=$C000
 30 C000              MEM           =   $FB
 40 C000 38                         SEC
 50 C001 A5FB                       LDA MEM
 60 C003 E930                       SBC #$30
 70 C005 C90A                       CMP #10
 80 C007 9002                       BCC SKIP
 90 C009 E907                       SBC #7
100 C00B 85FB         SKIP          STA MEM
110 C00D 60                         RTS
```

You should try out this trick with various combinations to convince yourself that it works every time.

What use is BCD?

Since BCD has been demonstrated to be inefficient, the obvious question is what do we get in return? One answer, not the most important, is simplicity. Only a cursory glance at the contents of a BCD byte is sufficient for most of us to convert into the equivalent decimal. However, a more substantial reason for including BCD in a computer is to make it compatible with external equipment, particularly the vast army of digitally controlled instruments. Most instruments of this form are equipped to either accept BCD formatted inputs or deliver BCD outputs. For example, a digital voltmeter can transmit readings from an external system to the computer in BCD 'language'. Conversely, a variable voltage power supply can be controlled by sending BCD information from the computer.

It is realised, of course, that the majority of readers may not have access to such equipment in which case these facilities may be of little interest. But, we should remember that any computer has vast potential. It is possible that addiction to games, entertaining though they may be for a time, may not last for ever and many users may feel the urge to exploit their machines in other directions. We should bear in mind that the input output bus in Commodore machines, ever since the days of the original PET 2001, have been based on an industry accepted standard known as the IEEE bus protocol. True, the Commodore employs a slightly modified version of the bus but, in general, it is reasonably compatible. We have no space in this article to discuss the details of the bus although we should point out that BCD is the accepted code used for passing numerical data between computer and external equipment.

BCD is also useful as a convenient

Program 12.5. Add two BCD Numbers and display result.

```
 10 033C            !ADD TWO BCD NUMBERS AND DISPLAY
 20 033C            !THE RESULT IN DEC DIGITS
 30 033C            NUMBER1        =   $FB
 40 033C            NUMBER2        =   $FC
 50 033C            RESULT         =   $FD
 60 033C            CHROUT         =   $FFD2
 70 C000            *=$C000
 80 C000 F8                        SED
 90 C001 A908                      LDA #8
100 C003 85FB                      STA NUMBER1
110 C005 A909                      LDA #9
120 C007 85FC                      STA NUMBER2
130 C009 18                        CLC
140 C00A A5FB                      LDA NUMBER1
150 C00C 65FC                      ADC NUMBER2
160 C00E 85FD                      STA RESULT
170 C010 2015C0                    JSR OUTPUT
180 C013 D8                        CLD
190 C014 60                        RTS
200 C015             !
210 C015 48          OUTPUT        PHA
220 C016 4A                        LSR A
230 C017 4A                        LSR A
240 C018 4A                        LSR A
250 C019 4A                        LSR A
260 C01A 18                        CLC
270 C01B 6930                      ADC #$30
280 C01D 20D2FF                    JSR CHROUT
290 C020 68                        PLA
300 C021 290F                      AND #$0F
310 C023 6930                      ADC #$30
320 C025 20D2FF                    JSR CHROUT
330 C028 60                        RTS
```

intermediate code useful as a stepping stone for other conversions.

Code conversions

The above discussion on BCD raises the general question of conversion between codes. It is often required to convert information from one code to another, either for reasons of efficiency or convenience. The conversion can be achieved by short program segments, preferably written in machine code. For the most part, they are short so instead of the usual practice of presenting an additional hex dump, we shall present them in full assembly form which includes the equivalent op-codes in hex.

Convert BCD digit to ASCII (See Program 12.1)

Program 12.1 Convert BCD digit to ASCII

For the benefit of those who do not have an assembler, the columns, reading from left to right, have the following significance:

Column 1 is a line number, used for reference pruposes only. (You can't jump to a line number like we do in BASIC).

Column 2 is the hex address of the location which stores the first byte of the current instruction.

Column 3 is the instruction in terms of machine code hex digits.

Depending on the instruction, there may be one, two or three pairs of hex digits. The first pair will always be the op-code and the remaining pairs, if any, will be a single or double byte operand.

Column 4 is the label field. That is to say, it will contain arbitrarily chosen labels for branch destinations. If no labels are used, the column is left blank.

Column 5 is the instruction in assembly language, consisting of the mnemonic code and operand.

Those without an assembler will have to place Column 3 machine code bytes in the data statements of the hex loading program given in part 3 of this series (December issue). Taking program 12.1 as an example, the data statements would be entered as follows:

DATA 18,A5,FB,69,30,85,FB,60

Description of program 12.1

Line 10 starts with ! so it is simply a remark.
Line 20 tells the assembler to locate the program starting at the address $C000. (In our assembler, $ means hex, not string).
Line 30 tells the assembler that the address $FB is to be known by the symbolic name MEM. It is assumed that the current contents of MEM is holding the BCD digit to be converted.
Line 40 clears the carry ready for the later addition.
Line 50 loads the accumulator with the BCD digit.
Line 60 uses immediate addressing to add hex 30 (decimal 48) to the BCD digit. Suppose MEM originally held 3. When 48 is added, it would hold 51 which is the ASCII code for 3.
Line 70 stores the converted number back in the original location. The program can be used as a subroutine called from within a BASIC program by using SYS 49152.

This simple program has been treated in extra detail because the full assembly format may be new to some readers. It is hoped that the programs which follow can be understood without resorting to a similar line by line analysis.

Converting ASCII to BCD digit (See Program 12.2)

Program 12.2 Converting ASCII to BCD digit

This is virtually the mirror image of the previous program, the only difference is that $30 is subtracted instead of being added. However, we should note that the carry has to be set, using SEC, before a new subtraction instead of being cleared by CLC.

Convert hex digit to ASCII (See Program (12.3)

Program 12.3 Convert hex digit to ASCII

Line 60 compares the value of the hex digit with 10 decimal. If the carry is clear in line 70, it signifies that it was less than 10 so a branch is made to line 90 which adds hex 30 to lift it into the ASCII band.

If however, the previous comparison

showed that the hex digit was 10 or greater, an extra six is added to allow for the gap of seven between the ASCII code for 9 and the ASCII code for A. Note that the carry is always set before the ADC 6 instruction so, in effect, we are adding 7.

Coverting ASCII to hex digit (See Program 12.4)

Program 12.4 Convert ASCII to the hex digit.

This program is almost a mirror image of the previous one, except of course that subtraction of hex 30 is involved instead of addition which is why line 40 sets the carry. Notice that, this time, the full 7 is subtracted (the gap between ASCII '9' and ASCII 'A') since the SBC instruction is entered with the carry set.

Addition of BCD numbers (See Program 12.5)

Program 12.5 Add two BCD numbers and display result

The program is intended only as a guide to the general procedure of adding in BCD. Because of this, the simple constants 8 and 9 have been used for the two numbers. The result, 17, is arranged to be printed out on the screen using the ROM subroutine CHROUT. (We must remember that CHROUT will only print the character corresponding to the ASCII code in the accumulator.) Practical versions will, of course, be more likely to supply the numbers as variables.

How the program works

Lines 30 to 60 assign the symbolic addresses NUMBER1, NUMBER2 and RESULT together with the ROM character output subroutine at address FFD2 hex.
Line 80, by use of the instruction SED, makes use of the 6502's ability to perform all following arithmetic in BCD.
Lines 90 and 120 store the arbitrary constants for later use.
Lines 130 to 160 performs the addition of BCD, the add-6 operation is carried out automatically by SED.
Lines 170 and 180 store the result and then call on OUTPUT. The machine is then reset for normal binary arithmetic by use of CLD. The subroutine OUTPUT is used to prepare the accumulator for action by CHROUT.

A copy of the raw data is first placed on the stack by use of PHA. The accumulator at this point will contain:

0001 0111 (17 BCD)

The accumulator is then shifted right 4 places in order to position the higher order digit at the right:

0000 0001

The addition of 30 hex is then added to lift the number into the ASCII band. the accumulator will now contain:

0011 0001 (31 hex, 49 decimal)

This is the ASCII for the digit '1' so when the subroutine CHROUT is called, this will appear as the first digit result on the screen. The original raw data is pulled back from the stack using PLA so the accumulator now holds, as before:

0001 0111 (17 BCD)

We now AND with:

0000 1111 (OF hex)

which produces:

0000 0111

As you can see, the accumulator now holds only the least significant digit (7) of the original BCD result. The usual 30 hex is then added to convert to ASCII so the accumulator now holds 37 hex (55 decimal) which is the ASCII code for 7. The accumulator is then sent to the screen via another call to CHROUT for displaying the second digit of the result.

Program 12.6. Displaying contents of location in hex.

```
 20 033C          !DISPLAYING THE CONTENTS OF A
 30 033C          !LOCATION IN HEX DIGITS
 40 033C          NUMBER       =  $FB
 50 033C          CHROUT       =  $FFD2
 60 C000          *=$C000
 70 C000          !
 80 C000 A5FB                LDA NUMBER
 90 C002 2006C0              JSR OUTPUT
100 C005 60                  RTS
110 C006          !
120 C006 48       OUTPUT     PHA
130 C007 4A                  LSR A
140 C008 4A                  LSR A
150 C009 4A                  LSR A
160 C00A 4A                  LSR A
170 C00B C90A                CMP #10
180 C00D 9002                BCC OVER
190 C00F 6906                ADC #6
200 C011 6930     OVER       ADC #$30
210 C013 20D2FF              JSR CHROUT
220 C016 68                  PLA
230 C017 290F                AND #$0F
240 C019 C90A                CMP #10
250 C01B 9002                BCC OVER2
260 C01D 6906                ADC #6
270 C01F 6930     OVER2      ADC #$30
280 C021 20D2FF              JSR CHROUT
290 C024 60                  RTS
```

Displaying contents of location in hex (See Program 12.6)

Program 12.6 Displaying contents of location in hex.

This program does not justify detailed explanation since it is broadly similar to the previous one. This time, SED is not used, we expect that somewhere along the line, we have to add six, (see line 170).

Summary

This article has attempted to show some of the various techniques which can be used for code conversion. Consequently, the programs should be interpreted only as guide lines to be introduced in practical subroutines.

ALL MIXED UP

This article from Gareth Thomas gives you an insight into mixing machine code and Basic.

IF YOU WRITE MACHINE CODE JUST TO improve your Basic programs, there comes a time when you M/C routines cannot stand alone and need values passed to them from Basic. The simplest way to achieve this is by POKEing the relevant values into memory and picking them out using your routine, if you do this the best locations to use are those used to store the 6502's registers which are accessed during a SYS call, they are as:

780 Accumulator
781 X register
782 Y register
783 Status register

Perhaps the best known example of this method is a basic program to PRINT AT using the KERNAL routine PLOT:

```
10  POKE 781,ROW
20  POKE 782,COLUMN
30  POKE 783,0
40  SYS 65520:REM call PLOT routine
50  PRINT "Pretty cumbersome though
    isn't it?"
```

Unfortunately, this method only allows you to use integers from 0-255.

If you need to use a bigger number a better method is to use the Basic USR function I've never seen this used in a program which is probably due to the lack of documentation about it in the manuals. The USR function enables the passing of one floating point number in the range 0-65535 to your m/c program. The function takes the value converts it to floating point and places it in the floating point accumulator 1 and then activates the M/C routine pointed to by the low/high bytes at 785/786. All your M/C routine needs is to use the routine CONV at $ B7F7 which converts a floating point number in FPACC11 to an integer in $14 and $15.

```
10  REM set USRADD to $CAAA
20  POKE 785,170
30  POKE 786,202
40  P=USR(65535):REM evaluates 65535
    to FPACC1 then executes routine at
    $CAAA
```

M/C routines:

```
CAAA  JSR   $B7F7   'convert FPA-
      LDA   $14     CC1 to integer
                    in $14 and $15
      TAX           'save low byte in
                    X-reg
      LDA   $15
CAB2  TAY           'save high byte
                    in Y-reg
```

The third method is slightly more complicated but is definitely the best, that is, extending the SYS call to pass values as well e.g. SYS addr, value 1, value 2, etc. This can only be done at machine level and needs the use of a number of ROM routines.

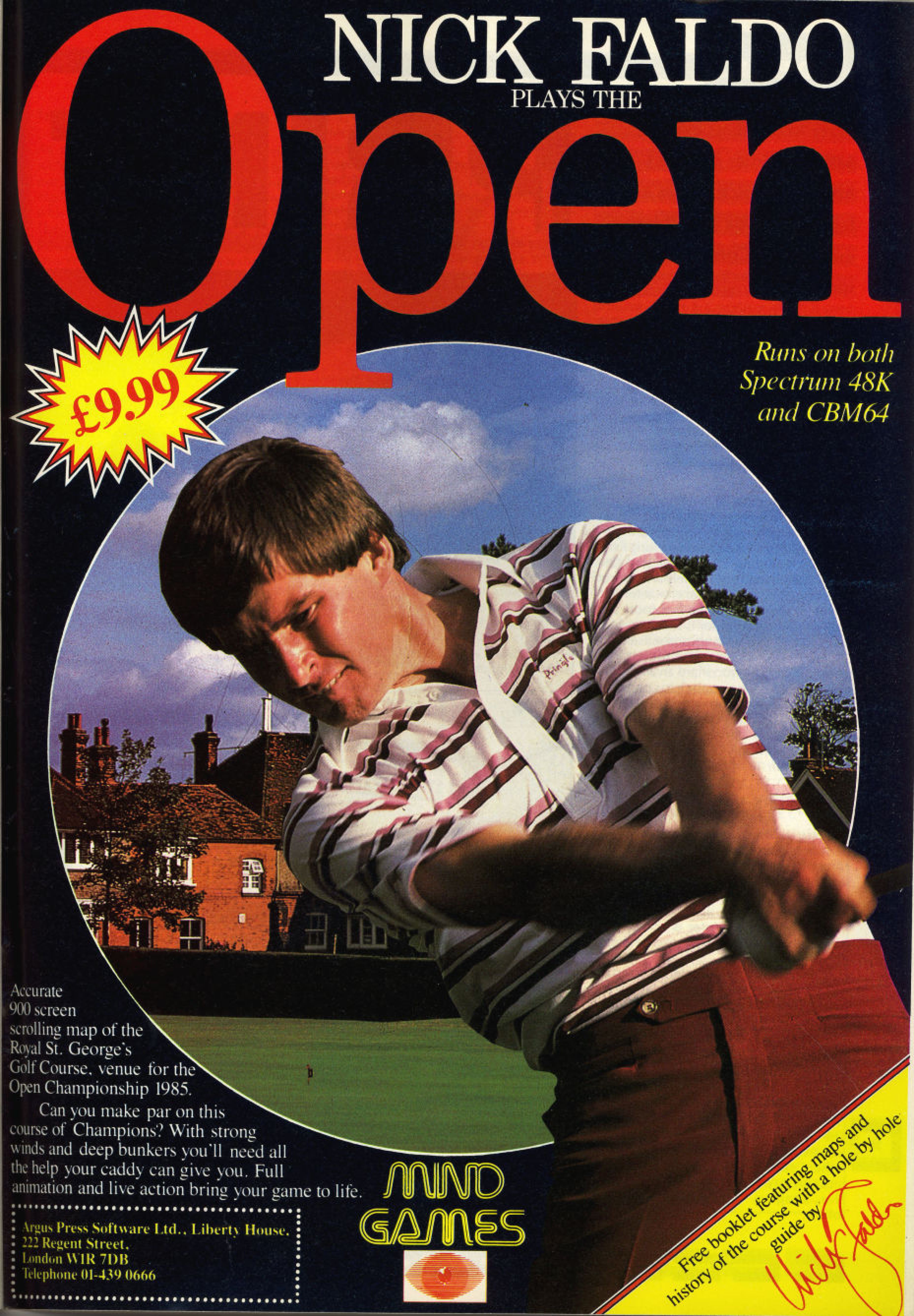
The first of these is CHARGET. Although this resides in ROM, a copy of it can be found located in ZERO-PAGE at location $0073. It is used by Basic to collect bytes of the current instruction (and is therefore in zero-page for speed), in fact we only need a part of it called CHARGOT at $0079. This collects the current byte without first updating the pointer, otherwise the point at $007B would be updated and we would miss the first parameter. The second routine is CEKCOM at $AEFD. This checks for a comma after the address, if it does not find one a 'SYNTAX ERROR' is generated. The third routine is EVAL at $AD8A this is a complex routine which evaluates an expression and converts it to floating point and then places it in FPACC1. The last routine CINT at $B7F7 I have already described.,

To illustrate how to use these routines I've written a small program which is a M/C version of listing 1 i.e. it sets the cursor position for PRINT but without all the POKEs, just one SYS call.

```
        JSR  $0079   'get byte
        JSR  $AEFD   'if not a comma
                     generate error
                     message
        JSR  $AD8A   'evaluate
                     expression
        JSR  $B7F7   'convert FPA-
                     CC1 to integer
                     in $14 and $15
        LDA  $14     'lead low byte
        CMP  # $28   '>than 40?
        BCC  NEXT    'no so get next
                     parameter
ERROR   JMP  $B248   'yes so generate
                     ILLEGAL
                     QUANTITY
                     error
NEXT    PHA          'save value on
                     stack
        JSR  $AEFD
        JSR  $AD8A
        JSR  $B7F7   'repeat as above
                     for second
                     paramter
        LDA  $14
        CMF  #$19    '>than 25
        BCS  ERROR   'yes so generate
                     error message
        TAX          'no so transfer
                     to X-register
        PLA          'retrieve last
                     value
        TAY          'transfer to Y-
                     register
        CLC          'clear carry (if set
                     PLOT will only
                     read cursor
                     position
        JSR  $FFF0   'execute PLOT
             routine
        RTS          'return to BASIC
```

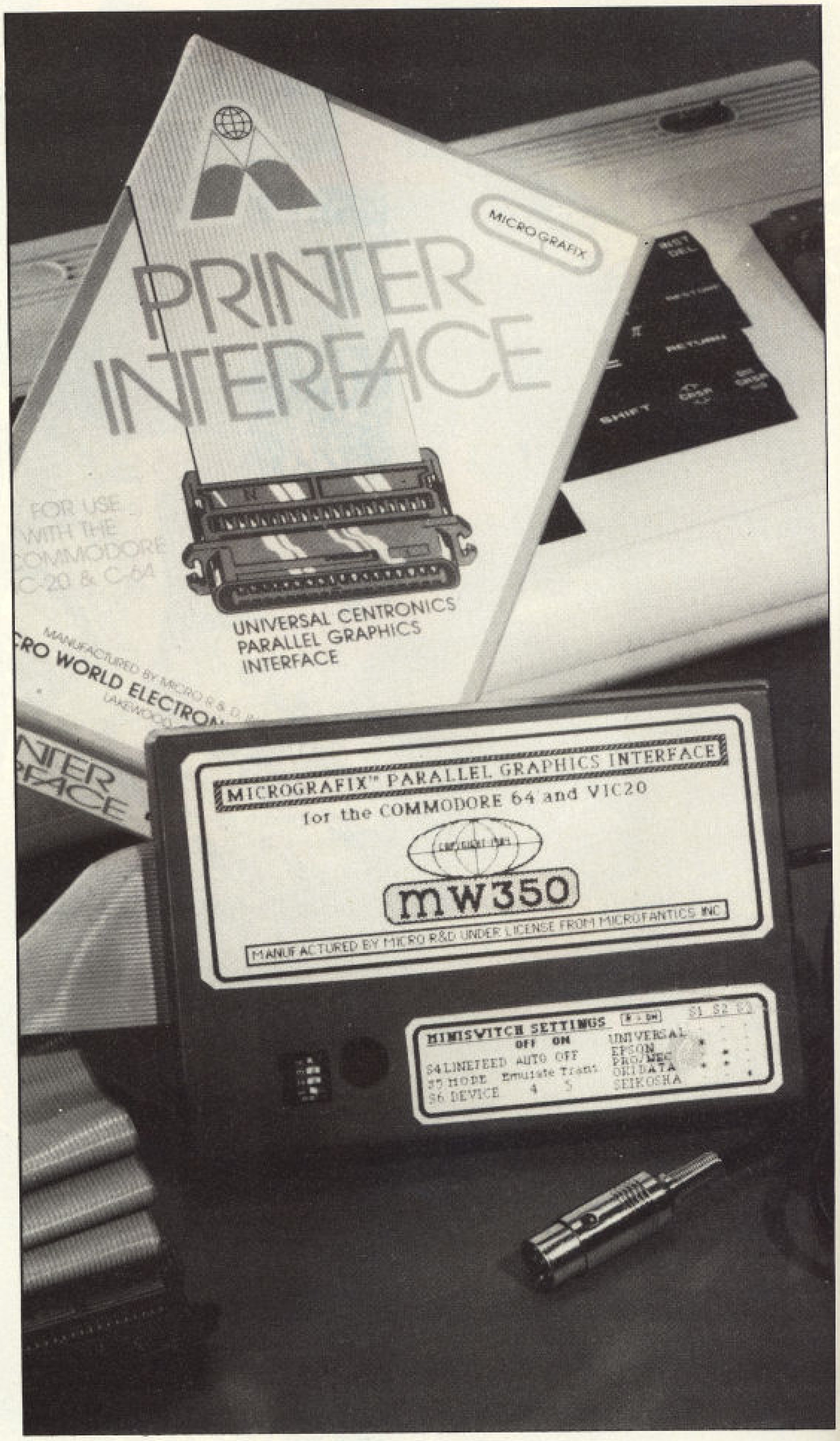
As you can see, the process is repeated for each paramter except that CHARGOT is only used once. Also, the advantage of using EVAL at $AD8A, is that expressions such as 5*COS(10) or variables can be used. Finally, to use the routine:

SYS addr,column,row:PRINT"message"


NICK FALDO
PLAYS THE
Open
£9.99
Runs on both Spectrum 48K and CBM64
Accurate 900 screen scrolling map of the Royal St. George's Golf Course, venue for the Open Championship 1985.
Can you make par on this course of Champions? With strong winds and deep bunkers you'll need all the help your caddy can give you. Full animation and live action bring your game to life.
MIND GAMES
Argus Press Software Ltd., Liberty House, 222 Regent Street, London W1R 7DB Telephone 01-439 0666
Free booklet featuring maps and history of the course with a hole by hole guide by Nick Faldo

IN PRINT

Barry Miles plugs into the MW350 Printer Graphics Interface, which makes a larger selection of printers available to Commodore users.



ADVANCED BUSINESS PROGRAMS FOR the CBM 64 use the entire memory of the machine. This entails conflict over the use of the same memory area. Micrographics' MW350 Printer Graphics Interface for the Commodore 64 and VIC 20 has its own power supply and, therefore, solves this problem!

Initial impressions

On the top surface of the attractive little box there is a reset button and six dip switches which enable you to configure your interface to match quite a long list of printers, including Epson, Nec and Oki. Other manufacturers' models are catered for where they have similar characteristics to one of those listed. The five-foot cable leading to the computer is sturdy, rubber covered and ends in a no-nonsense metal Din connector with a spring protecting it from being bent at too sharp an angle. The ribbon cable is long enough to connect to a fairly distant printer.

The additional connection to the cassette port is optional, because some printers will supply a five volt electrical supply to pin 18 of the 36 pin connector.

Connecting up

You are advised to ensure that all the peripherals are powered up before the computer is switched on and sends a reset signal to initialise the disc and printer. If the printer scrolls one line upwards, the interface is working. If it does not, you may find that the printer is not on line, possibly because the paper-end detection device is working.

Conveniently, the printer interface will produce a status report to show you what's what. The status report is quite informative. For example, it tells you which printer you have set your interface to work with, or whether your line feed is in operation.

Documentation

The documentation is very thorough. However, it is printed in green and in rather small type! A very detailed introductory section runs to six chapters. Five chapters of reference material follow.

In operation

As supplied the MW350 will function with virtually all printers. To get the best out of your system, you set up the three dip switches to match the printer type which you are using. I used an Epson FX80. The Universal setting will enable you to use a Daisywheel printer, although of course the graphics capability is then lost.

Switch 5 controls the ASCII character conversion, Switch 6 selects the device number. If you change the switch settings after power-up you must press the Reset button.

Commodore 1525 emulation

When dipswitch number 5 is in the off position, you obtain an exact simulation of the 1525's performance. If you send the appropriate control codes you will be able to do double-width printing, or have columnar tabs up to 80 print positions available by sending CHR$(16) followed by 2 numbers which specify the start column. You can also set up your own user-definable character and can even do dot tabulation, taking your print head to any one of 480 dot positions. There is a Graphic Repeat function which repeats a single graphics character.

In the *Emulate mode*, you are also able to use additional commands which are peculiar to the MW350. By opening a file to your printer with a secondary address of 1, you arrange that all the commands which normally result in a reverse field character appearing when a BASIC program is listed, are now translated into readable mnemonics. This makes your listings transparently obvious, even if you have not memorised the Commodore symbols.

In this mode you are not able to print Commodore graphics as graphics. What is printed instead is the key which would be struck on the keyboard in order to produce that graphic symbol. This is particularly helpful for printing out programs which use vertical or horizontal bar graphics symbols, which are very difficult to distinguish from one another.

You can also set the left and right hand printer margins to leave plenty of space for notes or to perforate the sheets for filing purposes.

Sending control codes

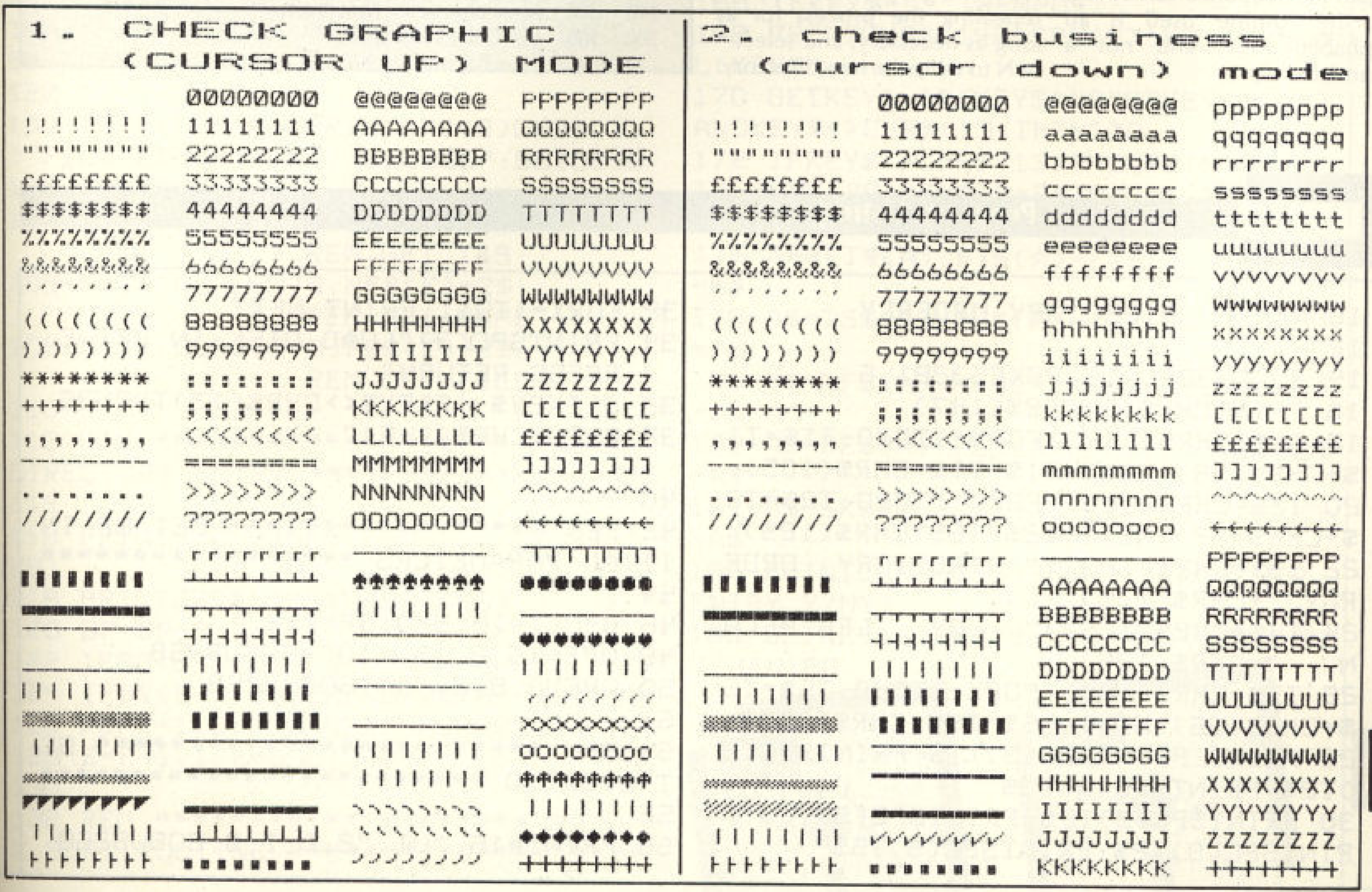
The *Transparent mode* of operation enables you to send codes to your printer which control various special features. You merely open the file to the printer with a secondary address of five. Once this is done all print commands are sent literally. Problems could arise if you wanted to send commands to the printer while you were already in Emulate mode. The Commodore 1525 itself reacts to certain escape sequences. However this has been catered for by arranging that the control sequence will be correctly transmitted to your printer if you precede it by an extra ESCAPE character.

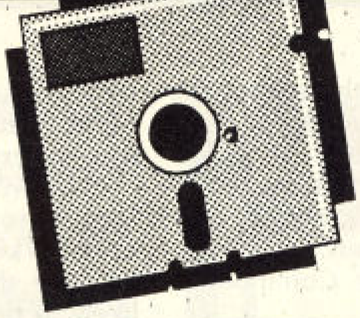
The MW350 has a number of extra features to make life easy. You will find it useful to keep the interface close at hand, because the switches and buttons are likely to be in continuous use. The push button initialises the interface. If you are using single sheet mode, the printer pauses at the bottom of each page. You simply press the button once to continue printing. There is a monitor which is used to check whether the interface is working satisfactorily. You can defeat the right margin of 80 columns if your printer has a wide platten and you need to use that.

There is a 'skip-over perforations' feature, which is disabled on power-up. This is combined with a form length setting command which enables you to accommodate different sized sheets of paper.

Conclusions

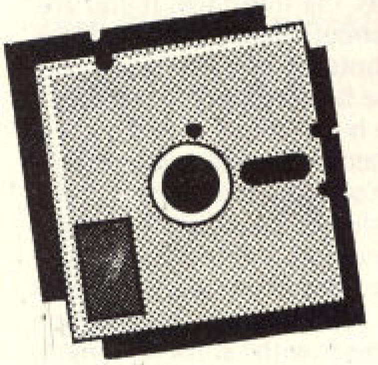
This interface has a wide range of capabilities, is very easy to use and fills a substantial need. Whilst quite expensive, its versatility, in offering Commodore users a wider selection of printers, justifies the price. And, the Emulate mode prevents the frustration of not being able to use Commodore printer-orientated software. The MW350 is well designed, robust and is highly recommended.





Are your files in a mess? Get them sorted out with this program by Les Allan.

D·I·S·K ORDERLY



WHEN SAVING FILES TO DISK, it is nice to retain an orderly sequence so as to keep a group of files together. However, when a file is scratched from the disk a 'gap' is left in the directory such that the next SAVE is made to that location. This means that if a number of associated files are being made the sequence is jumbled.

Utilities exist on the market such as a disk dissector that enables a tidy routine to be applied to the directory but again the sequence can be lost as the routine used is an alphabetic sort and not a manual one.

With this utility the order can be decided upon by the user simply by swapping any two locations at a time until the desired sequence is obtained. OK! It may take several operations to get the directory sequence but it works and works well.

The utility works by reading the directory contents into several arrays to store the name, track and sector, file type and file data which is then displayed on the screen for editing. Due to the limitation of the screen the maximum number of entries allowed is 40. Simply select the two locations to be swapped, repeating the process for as long as necessary, and select * or N to write the new directory.

Functional Listing

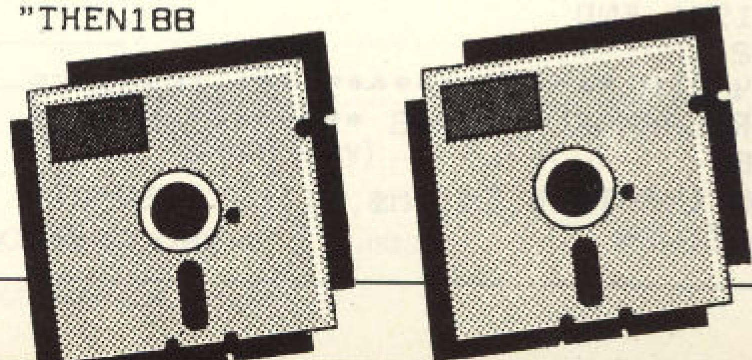
14 - 16	clear screen and set up colours
18 - 26	variables for screen title
28 - 38	print screen and look for RETURN key pressed
46	sets initial variables for track/sector/buffer pointer
48	initialises drive
50	opens a random file
58 - 70	get name from track 18/sector 0 and print to screen
78 - 82	dimensions arrays
90 - 108	read each directory entry and check for an open file
116 - 128	print each name to the screen
142 - 196	manual sort routine
204 - 260	write new directory to disk
268 - 270	disk error channel test
278 - 282	number of characters to be read from buffer
290 - 292	read character fom buffer
300 - 306	get OK to continue

Program Listing

```
10 REM     DIRECTORY ORDERLY
12 :
14 POKE53280,15:POKE53281,6
16 PRINTCHR$(5)CHR$(147)
18 T1$=CHR$(117):FORX=1TO20:T1$=T1
$+CHR$(96):NEXT:T1$=T1$+CHR$(105)
20 T2$=CHR$(125):FORX=1TO20:T2$=T2
$+CHR$(32):NEXT:T2$=T2$+CHR$(125)
22 T3$=CHR$(125)+" DIRECTORY  ORDE
RLY "+CHR$(125)
24 T4$=CHR$(125)+"   BY   LES ALLA
N   "+CHR$(125)
26 T5$=CHR$(106):FORX=1TO20:T5$=T5
$+CHR$(96):NEXT:T5$=T5$+CHR$(107)
28 PRINT:PRINTSPC(9)T1$:PRINTSPC(9
)T2$:PRINTSPC(9)T3$
30 PRINTSPC(9)T2$:PRINTSPC(9)T4$:P
RINTSPC(9)T2$:PRINTSPC(9)T5$
32 FORX=1TO11:PRINT:NEXT
34 PRINTSPC(3)"LOAD DISK IN DRIVE
   PRESS RETURN"
36 GETKEY$:IFKEY$<>CHR$(13)THEN36
38 PRINTCHR$(145)"
                                 "
40 :
42 REM **************** SET UP IN
ITIAL PARAMETERS ****************
44 :
46 D=0:T=18:S=0:BP=144
48 OPEN15,8,15,"IO":GOSUB268
50 OPEN2,8,2,"#":GOSUB268
52 :
54 REM ************************ GE
T DISK ID ***********************
56 :
58 PRINT#15,"U1:"2;D;T;S:GOSUB268
```

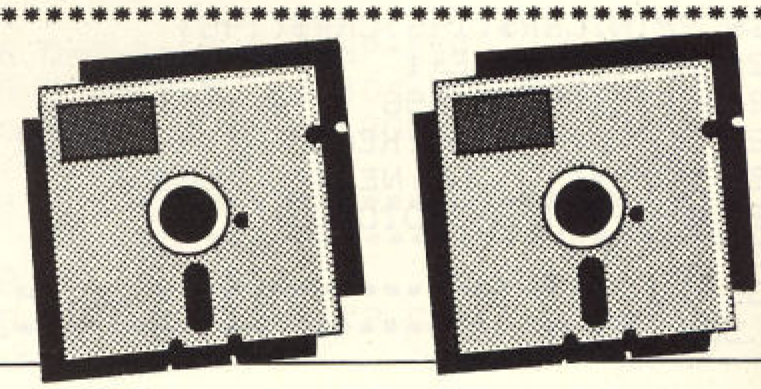
Program Listing (cont.)

```
60 PRINT#15,"B-P:";2;BP
62 ID$="":N=23:GOSUB272:ID$=Z$
64 PRINTCHR$(19):FORX=1TO11:PRINT:
NEXT
66 PRINTSPC(11)"DISK NAME  ID DOS"
68 PRINT:PRINTSPC(8)ID$
70 PRINT:PRINT:PRINT
72 :
74 REM ****************** SET UP D
IRECTORY ARRAYS ******************
76 :
78 DIMS%(19),FT$(192),TS$(192),NF$
(192),FE$(192),SK$(192),T$(4)
80 T$(1)="SEQ":T$(2)="PRG":T$(3)="
USR":T$(4)="REL"
82 FORI=1TO30:ZO$=ZO$+CHR$(0):NEXT
84 :
86 REM ****************** READ DIR
ECTORY ENTRIES ******************
88 :
90 S=1:E=1
92 PRINT#15,"U1:"2;D;T;S:GOSUB268:
BP=1
94 PRINT#15,"B-P:";2;BP:GOSUB290:S
%(S)=A:          REM GET NEXT SECTO
R
96 PRINT#15,"B-P:";2;BP:GOSUB284:F
T=A:             REM GET FILE TYPE
98 IFFT=0ORFT=128THENBP=BP+31:GOTO
122
100 IFFT<128THENPRINTSPC(8)CHR$(11
)"FILE ERROR   NOT CLOSED":GOTO252
102 FT$(E)=A$
104 N=2:GOSUB278:TS$(E)=Z$:
            REM GET T&S
106 N=16:GOSUB278:NF$(E)=Z$:
            REM GET NAME
108 N=11:GOSUB278:FE$(E)=Z$:
            REM GET FILE
110 :
112 REM ****************** PRINT
DIRECTORY ENTRY ******************
114 :
116 PRINTSPC(8)NF$(E)"   :  "T$(FT-1
28)
118 PRINTCHR$(145)CHR$(145)
120 BP=BP+2:E=E+1
122 IFBP<256GOTO96
124 IFS%(S)<>255THENS=S%(S):GOTO92
126 FORX=1TO1000:NEXT:GOSUB300
128 E=E-1:IFE=0GOTO204
130 :
132 REM ******************** MANUA
L SORT ROUTINE ******************
134 :
136 IFE>40THENGOTO194
138 PRINTCHR$(147)
140 FORI=1TOINT(E/2+.5):PRINTRIGHT
$(STR$(I),2)" "NF$(I):NEXT
142 PRINTCHR$(19)
144 FORI=INT(E/2+.5)+1TOE:PRINT,,R
IGHT$(STR$(I),2)" "NF$(I):NEXT
146 PRINTCHR$(19):FORX=1TO21:PRINT
:NEXT
148 PRINTSPC(3)"SWAP OVER WHICH NU
MBERS    * TO END"
150 PRINTSPC(11)"FROM ..       TO .
."
152 GETKEY$:IF(KEY$<"0"ORKEY$>"9")
ANDKEY$<>"*"THEN152
154 IFKEY$="*"THEN204
156 PRINTCHR$(145)SPC(16)KEY$:A=VA
L(KEY$)
158 GETKEY$:IF(KEY$<"0"ORKEY$>"9")
ANDKEY$<>CHR$(13)THEN158
160 IFKEY$<>CHR$(13)THENPRINTCHR$(
145)SPC(17)KEY$:A=A*10+VAL(KEY$)
162 IFA<1ORA>ETHEN146
164 GETKEY$:IF(KEY$<"0"ORKEY$>"9")
ANDKEY$<>"*"THEN164
166 IFKEY$="*"THEN204
168 PRINTCHR$(145)SPC(27)KEY$:B=VA
L(KEY$)
170 GETKEY$:IF(KEY$<"0"ORKEY$>"9")
ANDKEY$<>CHR$(13)THEN170
172 IFKEY$<>CHR$(13)THENPRINTCHR$(
145)SPC(28)KEY$:B=B*10+VAL(KEY$)
174 IFB<1ORB>ETHEN146
176 A$=FT$(A):FT$(A)=FT$(B):FT$(B)
=A$
178 A$=TS$(A):TS$(A)=TS$(B):TS$(B)
=A$
180 A$=NF$(A):NF$(A)=NF$(B):NF$(B)
=A$
182 A$=FE$(A):FE$(A)=FE$(B):FE$(B)
=A$
184 A$=SK$(A):SK$(A)=SK$(B):SK$(B)
=A$
186 PRINTCHR$(145)SPC(11)"ANOTHER
ENTRY Y/N?";
188 GETKEY$:IFKEY$<>"Y"ANDKEY$<>"N
"THEN188
```



Program Listing (cont.)

```
190 IFKEY$="Y"THEN138
192 GOTO204
194 PRINTCHR$(145)SPC(7)"TOO MANY
FILES FOR MANUAL":FORX=1TO2000:NEX
T
196 GOTO252
198 :
200 REM ****************** WRITE D
IRECTORY ENTRIES *****************
202 :
204 PRINTCHR$(147)"WRITING ... : "
ID$:PRINT
206 Z$=CHR$(0)+CHR$(255)
208 S=1:N=1
210 PRINT#15,"U1:"2;D;T;S:GOSUB262
212 BP=2
214 IFN>EGOTO230
216 IFBP<256GOTO222
218 PRINT#15,"U2:"2;D;T;S:GOSUB262
220 S=S%(S):GOTO210
222 PRINT#15,"B-P:";2;BP
224 PRINTSPC(14)NF$(N)" : "T$(ASC(
FT$(N))-128)
226 PRINT#2,FT$(N);TS$(N);NF$(N);F
E$(N);
228 BP=BP+32:N=N+1:GOTO214
230 IFBP>255GOTO238
232 PRINT#15,"B-P:";2;BP
234 PRINT#2,ZO$;
236 BP=BP+32:GOTO230
238 BP=0:PRINT#15,"B-P:";2;BP
240 PRINT#2,Z$;
242 PRINT#15,"U2:"2;D;T;S:GOSUB262
244 Z$=LEFT$(ZO$,2)
246 S=S%(S):IFS=255GOTO252
248 PRINT#15,"U1:";2;D;T;S:GOSUB26
2
250 BP=2:GOTO232
252 FORX=1TO1000:NEXT:CLOSE2:CLOSE
15
254 PRINTCHR$(147):FORT=1TO11:PRIN
T:NEXT:PRINTSPC(11)"ANOTHER DISK
Y/N?"
256 GETKEY$:IFKEY$<>"Y"ANDKEY$<>"N
"THEN256
258 IFKEY$="Y"THENRUN
260 POKE53280,14:PRINTCHR$(147)CHR
$(154):END
262 :
264 REM ***************** INPUT ER
ROR CHANNEL STATUS ***************
266 :
268 INPUT#15,EN,EM$,ET,ES:IFEN=0TH
ENRETURN
270 PRINTCHR$(147)"DISK ERROR "EN;
EM$;ET;ES:GOTO252
272 :
274 REM ***************** GET CHAR
ACTERS FROM BUF. *****************
276 :
278 Z$=""
280 FORI=1TON:GOSUB290
282 Z$=Z$+A$:NEXT:RETURN
284 :
286 REM ********************* INPU
T FROM BUFFER ********************
288 :
290 GET#2,A$:IFA$=""THENA$=CHR$(0)
292 A=ASC(A$):BP=BP+1:RETURN
294 :
296 REM ******************** GET O
K TO CONTINUE ********************
298 :
300 PRINTSPC(8)CHR$(11)"OK TO CONT
INUE     Y/N?"
302 GETKEY$:IFZ$="N"GOTO252
304 IFKEY$<>"Y"GOTO302
306 RETURN
308 :
310 :
312 **********************************
**********************************
314 *
                                 *
316 *           DISK DIRECTORY ORD
ERLY FOR COMMODORE 64            *
317 *
                                 *
318 *           SWAP ANY 2 DIRECTO
RY ENTRIES AT  A TIME            *
319 *           MAXIMUM NUMBER OF
ENTRIES ALLOWED :- 44            *
320 *
                                 *
321 *           LES ALLAN
         19.01.85                *
322 *
                                 *
324 **********************************
**********************************
```



HARDWARE

COMMODORE C16

DUAL CASSETTE INTERFACE

Transfer any C16 cassette program, turbos, m/code etc. to a second datasette.

Interface I for C16 and +4: **£11.95** inc P&P. Interface II for C16, +4, CBM 64 and Vic 20: **£14.95** inc P&P.

Joystick Interface, connects any joystick to C16: **£3.95** inc P&P.

Cassette Interface, connects any C2N cassette to C16: **£3.25** inc P&P.

Cheques/POs to:

J. FITZPATRICK (YC), 1 Alveley Ave, Withington, Manchester M20 9UD.



SOFTWARE GAMES

CBM 64

Soccer Supremo, soccer management game (HCW *****) : Pirate Island, board game simulation (HCW *****) : Cricket, limited over test match (HCW ****). £4.99 each. 2 for £8.99 or 3 for £12.50 on casette. Add £2.50 to total for disc.

From: Dromeda Software, 56 Wells Street, Haslingdon, Lancashire, BB4 5LS.

Super Breaker Back-Up Board

For the **COMMODORE 64** and **VIC-20**

When a second CBM type cassette unit is connected to the board it will make security back-up copies of cassette based software, including Turbo/Fast Loaders, Basic and/or Machine Code and Data Files. No additional software is needed. Very User Friendly, the back-up copy is made on the second cassette while the original is laoding on the first. Price at £8.95 plus 55p for p&p UK (Europe add £1.00, elsewhere add £2.00). Send cheque/postal order, or a stamp for details, to:

C. Kent, 20 Northcape Walk, Corby, Northants.

For enquiries telephone **(0536) 742622** between 3pm and 5pm, Monday to Friday.

DON'T BE AN ALIEN TO INVADE THIS SPACE RING 01 437 0699

ACCOUNTS SOFTWARE

CASH TRADERS' BOOK KEEPING

Purchase Day Book.
Daily Takings.
VAT Return (A,B,C,D,F).
Bank Statement.
Petty Cash Statement.
Detail listing of up to 99 nominal a/cs.
19 accounts for creditor control.
Final Accounts.

£199.00 (inc. VAT and postage)

Demonstration Pack: £11.50

(refundable against purchase)

CASH WITH ORDER TO:

Ref: YC, Quick-Count Ltd., 15 Neeld Cres., London NW4 3RP. 01-202 5486

(ans mc.)

YOUR COMMODORE — CLASSIFIED ADVERTISEMENT — ORDER FORM

ADVERTISERS PLEASE ADD 15% VAT

Advertise nationally in these columns for only 40p per word (min charge 15 words). Simply print your message in the coupon and send with your cheque or postal order made payable to **Argus Specialist Publications Ltd** to:

CLASSIFIED DEPARTMENT
YOUR COMMODORE
No 1 Golden Square, London W1
Tel: 01-437 0699

Name.......................................

Address....................................

....

....

Tel no (day)..............................

Please place my advert in **YOUR COMMODORE** for .... issues.

Please indicate no. of insertions required.

YOUR COMMODORE

Lineage: 40p per word.
Semi display: £9.00 per single column centimetre
Ring for information on series bookings/discounts.

All advertisements in this section must be prepaid.
Advertisements are accepted subject to the terms and conditions printed on the advertisement rate card (available on request).

01-437 0699
EXT. 322

Send your requirements to:
DAVID HORNE
ASP LTD, 1 GOLDEN SQUARE, LONDON W1

SOFTWARE

VIC-20 USERS

A renumber facility for your basic programs requiring no memory expansion. Cassette £2.00 including post and package.
Dept 1, Bolling Software, PO Box 42, Bradford, West Yorkshire, BD4 7UD.

ADDRESS BOOK CBM 64

This user friendly fully menu driven program is ideal for keeping tabs on addresses, telephone numbers, birthdays, club memebers, etc. Produce Christmas card/Birthday club lists etc. with multiple printout options. Also compatible with 1520 p/plotter. Only **£6.90** incl. From:
FINSOFT, 7 Barlow Road, Wilmslow, Cheshire SK9 4BE.



GRAND PRIX CBM 64. A program to call up and view Grand Prix results and automatically update driver and construct championship tables. Data tape included with latest results entered for 1985. £4.85. From J. Hudson, 32 Garrison Lane, Chessington, Surrey KT9 2LB.

FOR SALE

GADGET 64 £5.95

This 100% machine code CBM64 cassette utility offers: Append, Delete, Renumber (including Goto, Gosub, etc.) Satisfyingly simple to use. Send your Name, Address & P/O or Cheque for £5.95 (P&P Free) to:
J&J GILMOUR LTD,
Dept YC985, 4 Copse Close, Liss, Hampshire, GU33 7EW, England.
Tel: 0730-893430.

FOR SALE

CBM 64 quality colour monitor style T.V. complete system* including good software library for sale. £350.00 O.N.O. Ring Bentley (0420) 23565 (Hants). *Tape based.

!VIC-20! with cassette, console, switchboard, RAM-pack, 75 games plus books for £250 inclusive. Ring 552-8512 for details.



SOFTWARE APPLICATIONS

HOME ACCOUNTS. Put your house in order! Probably home computing's best use! Comprehensive coverage of bank accounts. Credit cards, HP inbuilt accuracy check. Records all transactions. Projects cashflow for any period ahead. Available for C.16 CBM64 or VIC-20. £8.45 or *free* details from Discus Software, Freepost, Windmill Hill, Brixham TQ5 9BR. Telephone 08045 55532.

FOR HIRE

COMMODORE C16 CASSETTE HIRE

AT LAST!. All the top games available for the C16 can now be hired from:

DABSOFT

Library constantly updated as new games become available. Membership fee of **£6.50** includes 2 FREE rentals!
OVER 50 TITLES
One price for any game. Charge of £1.25 inc P&P BOTH WAYS.
For complete memebership kit post cheque or postal order to:

DABSOFT (YC)
39 Guildford Road, Levenshulme M19 3FG. Tel: 061 445 4375.

COMMODORE 64 cassette library. Hire latest games and educational software, only one price £1.50 per cassette 7 days hire. Details S.A.E. pleae: C.C.L., 19 Darrell Way, Abingdon, OX14 1HQ.

VIC-20 AND CMB-64 SOFTWARE HIRE

300 titles. Free membership. Send 2 × 17p stamps for your hirekit to:
VIC-20 SOFTWARE HIRE (Y.M.), 8 Thurland Street, Nottingham.
PLEASE STATE MACHINE.

Attention! Commodore Users.

THERE'S A WHOLE LIBRARY OF SOFTWARE FOR YOU AT THE MIDLAND COMPUTER LIBRARY.

LIFE MEMBERSHIP – JUST £6!

HIRE FIRST TWO GAMES FREE!

- Choose from the latest titles on cassette and disc
- Hire for just £1.25 inc. p+p for 7 days
- Hire as many games as you want
- All games are original with full documentation
- Business and educational software too
- Fast 'return-of-post' service
- Low prices if you want to buy
- 14 day money back guarantee
- Over 1000 games in stock including US imports

MIDLAND COMPUTERS

Please send a cheque or postal order for £6.00 stating your machine, or send a large S.A.E. to:
Midland Computers Limited,
28 College Street, Worcester WR1 2SL Telephone: (0905) 611072

REPAIRS

REPAIRS

by the Arcade Experts to Atari, Vic 20, Commodore 64, Spectrum ZX81, TRS80, Dragon, T199/4A, BBC, plus associated peripherals
We charge a low hourly rate and many repairs cost as little as **£5.00** with a maximum of **£23.00** inc of VAT. P&P and insurance.
3 months guarantee parts and labour; 24hr turnround.
SEND NO MONEY NOW
Just your faulty unit with description of condition.

ZILEC ELEXTRONICS LTD,
Unit 8, Trent Industrial Estate., Wetmore Road, Burton upon Trent, Staffordshire.
Telephone Burton (0283) 33535.

COMMODORE REPAIRS

Commodore repairs by Commodore approved engineers. Repair prices: Vic 20 modulator **£8.05**, Vic 20 **£15.00**, CBM 64 from **£20.00**, C2N from **£9.00**, 1541 from **£20.00**, C64 psu **£13.50**, Spectrum **£16.50**, Dragon **£32.00**, Electrons, Atari.
For more information SAE or telephone:
G.C. BUNCE & SON,
36 Burlington Road, Burnham, Bucks SL1 7BQ.
Tel: (06286) 61696.

COMPUTER REPAIRS IN SOUTHEND

CBM 64, Vic 20, Spectrum, BBC, Electron.

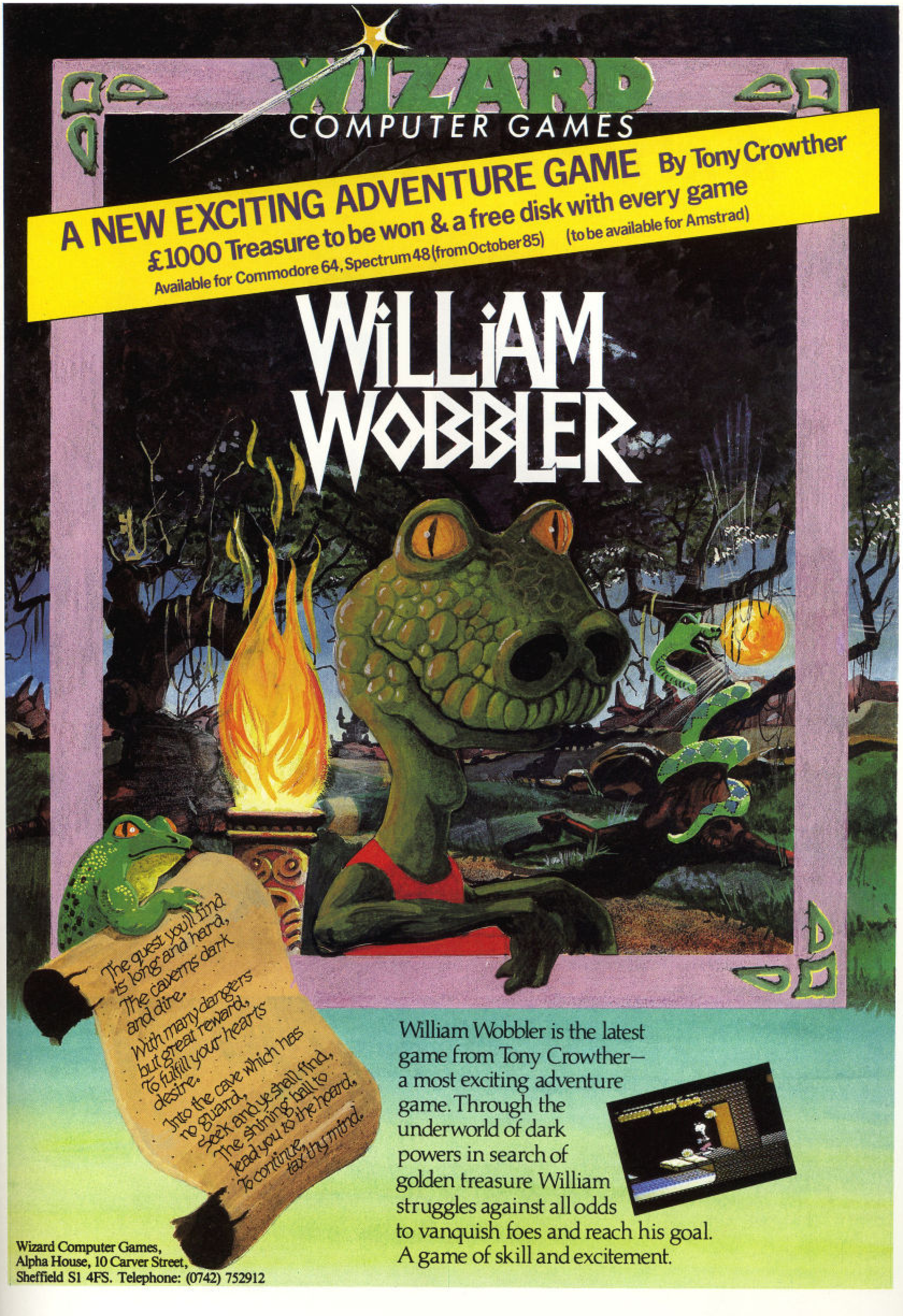
MINNIE MICRO ELECTRONICS
12 Eastern Esplanade, Southend.
Tel: 0702 62033/615809
Open 7 days a week

FOR SALE

MOONRAKER

CBM 64 Space Game features fast
- Fly-Into Rolling Radar Display
- Infra Red Target Screen

Altitude, Time, Score, Indicators, Fuel/Purchase Option, Promotion.
Send your name, address & PO/cheque to:
J & J Gilmour Ltd., Dept YC985, 4 Copse Close, Liss, Hampshire GU33 7EN, England GU33 7EN.
Tel: 0730-893430



WIZARD
COMPUTER GAMES
A NEW EXCITING ADVENTURE GAME By Tony Crowther
£1000 Treasure to be won & a free disk with every game
Available for Commodore 64, Spectrum 48 (from October 85) (to be available for Amstrad)
WILLIAM WOBBLER
The quest you'll find is long and hard, The caverns dark and dire. With many dangers but great reward, To fulfill your hearts desire. Into the cave which has no guard, Seek and ye shall find, The shining ball to lead you to the hoard, To continue, tax thy mind.
William Wobbler is the latest game from Tony Crowther– a most exciting adventure game. Through the underworld of dark powers in search of golden treasure William struggles against all odds to vanquish foes and reach his goal. A game of skill and excitement.
Wizard Computer Games, Alpha House, 10 Carver Street, Sheffield S1 4FS. Telephone: (0742) 752912

Choosing the right computer is a good start — but can you find the right software?



At SUPERSOFT we're very conscious of the fact that people who spend several hundred pounds on computer equipment are looking to do rather more than play Space Invaders.

Financial planning is a rather grand name for something you've been doing all your life – making ends meet! Perhaps if Mr Micawber had used **BUSICALC** he would have been able to balance the books a little better.

For home, club or small business use BUSICALC 1 should pay for itself in no time at all; for larger companies we recommend BUSICALC 3, one of the few really valuable programs that you can learn to use in a day.

Although your Commodore 64 is a powerful musical instrument you need to be a pretty good programmer to understand how it all works. Unless, of course, you buy **MUSIC MASTER!**

To use MUSIC MASTER requires no prior musical knowledge, though in the hands of an experienced musician it will prove an invaluable tool. You don't need to know the first thing about programming either! MUSIC MASTER is the musical equivalent of a word processor, remembering the notes you play and allowing you to replay and edit them as you wish.

INTERDICTOR PILOT is a space flight simulator. Nowadays simulators are widely used to train pilots and astronauts because – to be frank – it's a lot cheaper (and safer) than the real thing!

Imagine, if you will, life in the 22nd century: space travel is commonplace, and on the outskirts of the galaxy the first war between civilizations is being fought. A shortage of trained pilots has prompted the Federation to develop a computer simulation that allows raw recruits to gain experience without paying for their mistakes with their lives. With the aid of your Commodore 64 you too can learn to pilot the Interdictor Mk 3 craft. But be warned – this is no game!

SUPERSOFT

SUPERSOFT, Winchester House, Canning Road, Wealdstone, Harrow, Middlesex HA3 7SJ
Telephone: 01-861 1166

Other SUPERSOFT products include the **MIKRO ASSEMBLER** cartridge, the only assembler that's ideal for beginners yet powerful enough for the professional (most of our competitors use it!). The **VICTREE** cartridge adds dozens of commands to Basic including toolkit aids and disk commands; or on disk there's **MASTER 64,** a really comprehensive package for the keen programmer.

Of course, we do also publish games programs, and with classics like **STIX, QUINX** and **KAMI-KAZE** in our range we are one of the market leaders. But we most enjoy coming up with the sort of programs that are going to be in use for months and years, not hours and days – the sort of programs that make you glad that you bought a computer – and glad that you bought SUPERSOFT!

You won't find SUPERSOFT products on the shelves of your local supermarket. But most specialist shops stock titles from our extensive range (and are prepared to obtain other programs to order). However you can also buy direct by sending a cheque (pre-paid orders are post free!), by calling at our offices, or over the telephone using your ACCESS card.